Kognitiiviset vinoumat ohjelmistotuotannossa

Lilja Kettunen

Helsinki 24.11.2019

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen osasto

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Studieprogram — Study Programme
Matemaattis-luonnontieteellinen tiedekunta	Tietojenkäsittelytiede

Tekijä — Författare — Author
Lilja Kettunen

Työn nimi — Arbetets titel — Title

Kognitiiviset vinoumat ohjelmistotuotannossa

Ohjaajat — Handledare — Supervisors
Fabian Fagerholm ja Tomi Männistö

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Pro gradu -tutkielma	24.11.2019	50 sivua + 0 liitesivua

Tiivistelmä — Referat — Abstract

Pro gradu -tutkielman tavoitteena oli selvittää kognitiivisten vinoumien eli systemaattisten ajatusvirheiden ilmenemistä ja vaikutusta ohjelmistotuotannossa. Kognitiiviset vinoumat ovat sisäsyntyinen ilmiö, jotka voivat heikentää ohjelmistokehittäjien päätöksentekokykyä ja siten vaikuttaa toteutettavan ohjelmiston laatuun. Toisaalta kognitiivisista vinoumista voi olla myös hyötyä; niiden ansiosta kehittäjät kykenevät tekemään nopeita ja intuitiivisia päätöksiä.

Tutkielma toteutettiin systemaattisena kirjallisuuskatsauksena. Tutkielmassa selvitettiin, millä tavoin kognitiiviset vinoumat voivat esiintyä ohjelmistotuotannon eri vaiheissa; vaatimusmäärittelyssä, suunnittelussa, toteutuksessa ja testauksessa. Tutkielmassa kartoitettiin myös keinoja, joilla kognitiivisten vinoumien oletettuja haitallisia vaikutuksia voitaisiin vähentää.

Kognitiivisilla vinoumilla havaittiin olevan tulosten perusteella useita eri vaikutuksia ohjelmistokehityksessä. Vinoumat voivat esimerkiksi vääristää työmääräarvioita, aiheuttaa ohjelmointivirheitä tai johtaa puutteelliseen testaamiseen. Keinoja vinoumien vähentämiseksi olivat tulosten mukaan koulutus, tekniset menetelmät sekä tehtävän tai ympäristön muokkaaminen niin, että päätöksentekijä käyttää tehtävään sopivia luontaisia ongelmanratkaisustrategioita.

Monet vinoumien vähentämiskeinot olivat vain ehdotuksia eikä niitä ollut tutkittu kokeellisesti, joten keinojen kehittämisen ja niiden tehokkuuden selvittämisen havaittiin vaativan lisää tutkimusta.

ACM Computing Classification System (CCS):
Applied computing → Law, social and behavioral sciences → Psychology

Avainsanat — Nyckelord — Keywords
ohjelmistotuotanto, kognitiivinen vinouma, kognitiivinen harha

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Ohjelmistojärjestelmien erikoistumislinjan tutkielma

Sisältö

1 Johdanto

Ohjelmistotuotanto on systemaattista ja loogista ohjelmistojen suunnittelua ja toteutusta, joka perustuu valtaviin määriin teknologista informaatiota ja jota harjoitetaan hyväksi havaittujen metodien avulla [1]. Ohjelmistot vaativat virheetöntä koodia toimiakseen oikein, ja ohjelmistokehittäjiltä vaaditaan luovaa ongelmanratkaisua ja terävää päättelykykyä monimutkaisten ja toimivien järjestelmien toteuttamisessa [2]. Ohjelmistot ovat loogisuuden ilmentymiä - mutta niitä suunnittelevat ihmiset eivät välttämättä ole [3].

Ohjelmistojen suunnittelu on monimutkaista ja ne sisältävät valtavasti informaatiota, jota voi olla mahdotonta sisäistää kokonaan [2]. Suunnittelu vaatii valtavasti ihmisen ajattelutyötä, jota joudutaan joskus tekemään intuitiivisesti ja puutteellisilla tiedoilla. Virheet päättelyssä voivat johtaa huonoihin valintoihin ja päätöksiin, jotka voivat heikentää ohjelmiston laatua tai haitata projektin etenemistä [4]. Ohjelmistoihin voi esimerkiksi jäädä enemmän virheitä ja työmääräarviot olla epärealistisia, mikä voi johtaa ohjelmistoprojektien viivästymiseen ja budjettien ylittymiseen [5].

Kehnot päätökset ohjelmistosuunnittelussa voivat johtua kognitiivista vinoumista, jotka ovat systemaattisia ja toistuvia virheitä ihmisen päättelyssä ja päätöksenteossa [3]. Vinoumat ovat kaikille ihmisille sisäsyntyisiä ominaisuuksia, joita ilmenee ennustettavasti eri tilanteissa. Esimerkiksi yksi tunnetuimmista kognitiivisista vinoumista on vahvistusvinouma, joka saa ihmiset etsimään omia uskomuksia vahvistavaa informaatiota ja hylkäämään tietoa, joka kyseenalaistaa niitä [6]. Kognitiiviset vinoumat auttavat ihmisiä tekemään nopeita ja intuitiivisia päätöksiä, mutta ne voivat myös johtaa irrationaaliseen päättelyyn.

Tämän tutkielman tavoitteena oli selvittää, miten kognitiiviset vinoumat voivat ilmetä ohjelmistotuotannossa ja millaisia vaikutuksia niillä voi olla ohjelmistokehityksessä. Koska vinoumilla voi olla haitallisia vaikutuksia ohjelmistotuotannossa, tavoitteena oli myös selvittää, voiko vinoumien vaikutusta jotenkin vähentää ja millaisin keinoin.

Vinoumien tunnistaminen ohjelmistotuotannossa on tärkeää, jotta voitaisiin havaita heikkouksia ohjelmistojen suunnittelussa. Mutta sitäkin tärkeämpää olisi löytää keinoja vinoumien vähentämiseksi, jotta voitaisiin oppia välttämään systemaattisia virheitä ohjelmistojen suunnitteluun ja toteutukseen liittyvässä päätöksenteossa.

Tutkimus suoritettiin kirjallisuuskatsauksena. Tutkielmassa tarkastellaan, mitä kognitiivisia vinoumia on tutkittu ohjelmistotuotannon näkökulmasta ja kartoitetaan alan tutkimuksen tämänhetkistä tilaa. Tutkielmassa selvitettiin myös, miten vinoumien vähentämistä on tutkittu ohjelmistotuotannon näkökulmasta ja mitä tutkitusti toimivia menetelmiä on löydetty. Aineiston laatua arvioitiin analysoimalla sitä, miten hyvin alkuperäistä psykologian tutkimusta artikkeleissa hyödynnettiin.

Tutkielma on jaettu kuuteen lukuun. Seuraavassa luvussa käydään läpi tutkielman teoreettinen tausta. Luvussa kerrotaan, mitä kognitiiviset vinoumat ovat, ja pyritään havainnollistamaan miksi ne syntyvät. Luvussa esitellään muutamia kognitii-

visia vinoumia ja kerrotaan, miten niitä voidaan kategorisoida eri tyyppeihin. Taustaluvussa kerrotaan myös, miten vinoumat voivat vaikuttaa ohjelmistotuotannossa ja millaisin keinoin niitä voidaan yrittää vähentää.

Kolmannessa luvussa selostetaan, miten tutkimus suoritettiin. Luvussa käydään läpi tutkimuskysymykset ja tutkielman tavoitteet. Menetelmäluvussa kerrotaan myös, miten tutkielman aineiston haku suoritettiin ja miten aineistoa analysoitiin.

Neljännessä luvussa kerrotaan tutkimuksen tulokset. Luvussa selostetaan, mitä vinoumia ohjelmistotuotannossa on havaittu ja mitä seurauksia niillä voi olla. Tuloksissa kerrotaan myös, mitä keinoja vinoumien vähentämiseksi on ehdotettu tai kehitetty. Lisäksi tarkastellaan, miten aineistossa on hyödynnetty alkuperäistä psykologian tutkimusta.

Aineistoa analysoidaan viidennessä luvussa. Luvussa pohditaan tutkimuksen tuloksia ja vastataan tutkimuskysymyksiin. Lopuksi pohditaan tutkimuksen rajoitteita.

Viimeisessä luvussa on tutkielman ja tutkimustulosten yhteenveto sekä niistä tehdyt johtopäätökset. Lopuksi pohditaan, minkälaista tutkimusta aihe vaatii lisää, miten hyvin tutkimus vastasi tavoitteisiin ja mitkä olivat tutkimuksen tärkeimmät havainnot.

2 Teoriatausta

Luvussa määritellään tutkielman aiheeseen liittyvät konseptit ja termit. Luku sisältää kognitiivisen vinouman määritelmän ja selityksen vinoumien taustamekanismeista. Lisäksi määritellään muutamia kognitiivisia vinoumia ja kerrotaan, miten niitä voi esiintyä ohjelmistokehittäjillä. Lopuksi kerrotaan, millaisin keinoin kognitiivisia vinoumia voidaan ehkäistä päätöksenteossa.

2.1 Kognitiivisen vinouman määritelmä

Kognitiiviset vinoumat ovat systemaattisia virheitä ihmisen arviointikyvyssä ja päätöksenteossa, jotka syntyvät kognitiivisten ”oikoteiden” käyttämisestä [7]. Kognitiivisia vinoumia kutsutaan myös kognitiivisiksi harhoiksi tai -vääristymiksi. Ne ovat sisäänrakennettuja, automaattisia ja väistämättömiä seurauksia ihmisen ajatteluprosesseista [8]. Kognitiiviset vinoumat voivat johtaa vääristyneeseen havainnointiin ja tulkintaan tai virheelliseen päättelyyn.

Kognitiiviset oikotiet yksinkertaistavat monimutkaisiin tehtäviin liittyvää päätöksentekoa [3]. Ne ovat erittäin hyödyllisiä, sillä niiden tarkoitus on saavuttaa ongelmaan tarpeeksi hyvä ratkaisu erittäin lyhyessä ajassa [8]. Usein nopeasti tuotetun riittävän hyvän ratkaisun tuottaminen onkin tehokkaampaa kuin optimaalisimman ratkaisun saavuttaminen, jos se johtaa mielekkääseen lopputulokseen. Optimaalisimman ratkaisun päättely saattaa vaatia paljon aikaa ja informaation käsittelyä, ja jos lähes yhtä hyvään lopputulokseen päädytään oikoteitä käyttämällä, ne säästävät aikaa ja kognitiivisia resursseja. Mutta joissakin tilanteissa nämä päätöksentekomekanismit eivät olekaan sopivia ja silloin ne johtavat ennustettaviin päättelyvirheisiin eli kognitiivisiin vinoumiin [8, 9].

Vinoumien syiksi on arveltu kognitiivisia rajoitteita muistissa ja ajattelukyvyssä, mukautumista luonnolliseen ympäristöön evoluution seurauksena, sekä motivoivia tekijöitä, kuten halua säilyttää positiivista minäkuvaa [7].

2.2 Vinoumien taustamekanismi

Psykologien mukaan on olemassa kaksi erilaista ajattelun muotoa; intuitiivinen ja reflektiivinen [10]. Näitä kutsutaan myös järjestelmiksi 1 ja 2. Ajattelun jako kahteen muotoon on psykologi Daniel Kahnemanin kehittämä malli, joka havainnollistaa ihmisen erilaisia tapoja prosessoida informaatiota. Jako järjestelmiin 1 ja 2 ei ole fysiologinen, vaan mallin tarkoitus on selkeyttää ymmärrystä ihmisen päätöksentekomekanismeista.

Järjestelmä 1 mahdollistaa ajattelun ja päätöksenteon nopeasti ja automaattisesti. Järjestelmän 1 (J1) ajattelu ei vaadi tietoisuutta. Ihmiset käyttävät tätä järjestelmää jokapäiväisten asioiden tekemiseen, joita voidaan tehdä ilman keskittymistä, kuten kävelyyn ja hampaiden pesemiseen [11]. J1 tuottaa tavallisesti intuitiivisia vastauk-

sia tai ”mututuntumia” sekunnin murto-osissa [8]. Järjestelmän 1 haittapuoli on se, että nopea ja tehokas ajattelu vaatii paljon informaation yleistystä ja mahdollisen tärkeän informaation laiminlyöntiä [8]. Kognitiivisten vinoumien uskotaan syntyvän järjestelmän 1 käytöstä, jolloin päätös tehdään intuitiivisesti ilman tietoista harkintaa [11].

Järjestelmän 2 ajattelu on hidasta, tarkoituksenmukaista ja vaatii henkistä ponnistelua. Ihmiset käyttävät tätä järjestelmää opetellessaan jotain uutta tai tehdessään vaativaa ongelmanratkaisua. Järjestelmän 2 (J2) käyttäminen vaatii aikaa, tietoista informaation käsittelyä ja motivaatiota, joten sen käyttäminen on kognitiivisesti kuormittavampaa kuin järjestelmän 1 käyttö [8].

Molemmat järjestelmät ovat jatkuvasti käytössä, mutta J1 on aktiivinen suurimman osan ajasta. J2 aktivoituu, kun tapahtuu jotain yllättävää, kuten esimerkiksi kun huomaamme selkeän virheen tai kun tarvitsemme sääntöihin perustuvaa päättelyä [11].

J1 tuottaa yleensä nopeasti tilanteeseen sopivia päätelmiä, mutta ne voivat olla joskus vääriä, mistä voi seurata muun muassa kognitiivisia vinoumia. Ihmiset eivät pysty havaitsemaan, milloin J1 tuottaa vääriä johtopäätöksiä, joten niiden tunnistaminen on hyvin vaikeaa, eikä kokemuksen kartuttaminenkaan kehitä niiden tunnistamista [11]. Toisaalta, vaikka järjestelmää 2 käytetään tekemään tietoisia ja harkittuja päätöksiä, myös se voi tuottaa huonoja ja jopa epäloogisia päätelmiä [10].

Ihmisten päätöksentekokykyyn voivat vaikuttaa esimerkiksi motivoivat ja itsekkäät tavoitteet: päätöksentekotilanteessa on houkuttelevaa tehdä valinta, joka jollain tavalla hyödyttää itseä. Päätöksentekijät saattavat myös kiintyä ratkaisuihinsa tai valintoihinsa, jolloin he ovat haluttomia harkitsemaan muita vaihtoehtoja. Ihmisillä on myös taipumus tuottaa ongelmaan yksi hypoteesi ja perustella omaa valintaansa etsimällä ainoastaan valintaa tukevia todisteita [11].

Kognitiivisten vinoumien uskotaan olevan seuraus päätöksentekoprosesseista, jotka ovat evoluution myötä mukautuneet ratkaisemaan ihmisen luonnollisessa ympäristössä kohtaamia ongelmia [12]. Tästä näkökulmasta katsoen ihmisen ajattelu on rationaalista, ja vinoumat eivät siis ole pohjimmiltaan vajavuuksia vaan adaptaatiota luonnollisiin olosuhteisiin [7].

Mukautumista luonnolliseen ympäristöön voidaan tarkastella esimerkiksi erään kognitiivisen vinouman, saatavuusharhan näkökulmasta. Saatavuusharhan takia viimeaikaiset ja usein esiintyvät tapaukset palautuvat helpommin mieleen kuin harvoin ja kauan sitten tapahtuneet asiat [3]. Erään näkemyksen mukaan muistia ei ole tarkoitettu ainoastaan informaation tallentamiseen, vaan sen tarkoitus on palauttaa mieleen sellaista relevanttia tietoa, josta on jotain hyötyä kyseisessä tilanteessa [7]. Tässä valossa saatavuusharha ei ole virheellistä ajattelua, vaan se toimii tilanteeseen nähden tarkoituksenmukaisesti. Joissakin tilanteissa saatavuusharhasta on kuitenkin haittaa, kun ihminen tyytyy helposti saatavilla olevaan muistinvaraiseen tietoon eikä ota huomioon muita vaihtoehtoja.

Myös itsevarmuusharhasta, joka saa ihmisen pitämään itseään todellista eteväm-

pänä, on ollut hyötyä luonnollisissa olosuhteissa. Olemalla liian itsevarma omista kyvyistään ihmiset pystyvät antamaan kilpailevalle lajitoverille todellista voimakkaamman ja pätevämmän vaikutelman itsestään [12]. Tällöin kilpailija luovuttaa todennäköisemmin. Kun ihminen uskoo välittämäänsä informaatioon, käyttäytyminen on uskottavampaa ja kognitiivisesti helpompaa [12]. Luonnollisessa ympäristössä itsevarmuusharhasta on täten ollut hyötyä selviytymisen kannalta muun muassa konfliktien välttelyssä.

Optimismiharha saa ihmiset uskomaan tulevien tapahtumien päättyvän positiivisesti. Vaikka ennustukset tulevista tapahtumista saattavatkin olla väärin, optimistisilla uskomuksilla on mahdollisuus johtaa parempiin lopputuloksiin kuin realistisilla uskomuksilla [13]. Tällöin ihminen päättää yrittää, vaikka todellisuudessa mahdollisuudet onnistua olisivat pienet.

Vaikka kognitiivisista vinoumista seuraa joissain tilanteissa vääriä johtopäätöksiä, ne ovat olleet ihmisen selviytymisen kannalta myös siis hyödyllisiä.

2.3 Kognitiivisia vinoumia

Kognitiiviset vinoumat voidaan jakaa eri kategorioihin niiden tyypin mukaan. Mohanani et al. [14] jakaa vinoumat kahdeksaan kategoriaan, jotka perustuvat Fleischmannin kognitiivisten vinoumien kategorioihin [15]. Vinoumat jaetaan intressivinoumiin, vakausvinoumiin, toimintavinoumiin, hahmontunnistusvinoumiin, havainnointivinoumiin, muistivinoumiin, päätöksentekovinoumiin ja sosiaalisiin vinoumiin.

Intressivinoumat vääristävät päätöksentekoa yksilön omien mieltymysten ja ideoiden suuntaan. Intressivinoumiin kuuluva vinouma on esimerkiksi vahvistusharha [14].

Vakausvinoumat saavat ihmiset pitäytymään olemassa olevissa tai tutuissa vaihtoehdoissa ja hylkäämään uutta informaatiota. Ankkurointivaikutus kuuluu vakausvinoumiin.

Toimintavinoumien takia ihmiset eivät harkitse kaikkea päätöksentekoon liittyvää informaatiota tai muita vaihtoehtoja, vaan tekevät ennenaikaisia päätöksiä. Itsevarmuusharha ja optimismiharha kuuluvat toimintavinoumiin [14].

Hahmontunnistusvinoumat saavat ihmiset keskittymään informaatioon, jossa on heille ennestään tuttuja piirteitä. Saatavuusvinouma kuuluu tähän kategoriaan [14].

Havainnointivinoumat vaikuttavat siihen, miten informaatiota kerätään ja tulkitaan. Esimerkki havainnointivinoumista on kehysvaikutus ja edustavuusharha [15].

Muistivinoumat vaikuttavat siihen, miten informaatio tai kokemukset palautuvat mieleen.

Päätöksentekovinoumat vaikuttavat haitallisesti päätöksenteon aikana päätöksen laatuun.

Sosiaaliset vinoumat vaikuttavat yksilön asenteeseen toisista ihmisistä.

Muutamia kognitiivisia vinoumia ja niiden määritelmät on esitetty taulukossa 1.

Vinouma	**Määritelmä**
Ankkurointivaikutus	*Ankkurointivaikutus* saa ihmiset olemaan riippuvaisia ensimmäisestä kohtaamastaan informaatiosta. Tällöin tämä alustava informaatio, tai lähtöpiste, vaikuttaa yksilön päättelyyn [9]. Eri lähtöpisteistä eli ”ankkureista” muodostetaan samaan kysymykseen erilaisia vastauksia, jotka ovat vääristyneitä lähtöarvon suuntaan.
Itsevarmuusharha	*Itsevarmuusharha* on omien kykyjen yliarvioimista ja vastustajien, tehtävien tai riskien aliarvioimista. Itsevarmuusharhan takia ihmiset luulevat olevansa taitavampia ja tietävänsä enemmän kuin todellisuudessa [16]. Ihmisillä on taipumus uskoa olevansa keskimääräistä parempia kuin muut, vaikka tämä on tilastollisesti mahdotonta. Vinouma saa ihmiset kokemaan hallinnan tunnetta ja vähättelemään riskejä.
Saatavuusharha	*Saatavuusharhan* takia viimeaikaiset ja usein esiintyvät tapaukset ovat helpommin palautettavissa mieleen kuin vanhemmat tapahtumat tai sellaiset tapaukset, joita esiintyy harvoin [3]. Ihmiset käyttävät tapauksen saatavuutta sen yleisyyden ja todennäköisyyden arviointiin. Jos tapaus ilmenee usein ja se muistuu mieleen nopeasti, niiden toistuminen arvioidaan tiheämmäksi ja todennäköisemmäksi verrattuna tapaukseen, jonka instanssit eivät palaudu mieleen niin helposti [3].
Vahvistusharha	*Vahvistusharha* saa ihmiset etsimään ja tulkitsemaan informaatiota tavalla, joka tukee omia käsityksiä, uskomuksia ja hypoteeseja [6]. Tällöin positiivisille, käsityksiä vahvistaville tapauksille annetaan enemmän painoarvoa kuin informaatiolle, joka osoittaa käsityksen vääräksi. Ihmisillä on taipumus jopa vältellä informaatiota, joka on vastoin heidän käsityksiään ja hypoteesejaan, tai joka tukee muita vaihtoehtoja [6].

Kehysvaikutus	Ongelmat on usein mahdollista esittää, tai *kehystää*, monella eri tavalla. Ongelman muotoilu vaikuttaa ongelman ja sen toimintojen ja seurausten tulkintaan. Kun sama ongelma esitetään eri näkökulmista, se vaikuttaa päätöksentekijän tulkintaan eri vaihtoehdoista [17]. Kun ongelma voidaan muotoilla eri tavoilla ja se vaikuttaa päätöksentekijän valintaan, ilmiötä kutsutaan *kehysvaikutukseksi*.
Ikea-efekti	*Ikea-efekti* saa yksilön yliarvioimaan itse tekemiensä esineiden tai asioiden arvon. Kun ihmiset joutuvat käyttämään paljon aikaa ja vaivaa jonkin asian tekemiseen, he alkavat pitää tuotostaan todellista arvokkaampana [18]. Ikea-efekti ilmenee vain silloin, kun työ on onnistunut ja saatettu loppuun. Jos työ tuhotaan valmistumisen jälkeen tai sen loppuunsaattamisessa epäonnistutaan, ikea-efektin vaikutus katoaa [18].
Not-invented-here -ilmiö	*Not-invented-here-* tai NIH-ilmiö on yrityksissä toimivien projektitiimien taipumus pitää sisäisesti toteutettuja tuotteita parempina kuin yrityksen ulkopuolelta tulevia ideoita ja teknologioita [19]. Yrityksen sisäisiä tuotteita suositaan silloinkin, kun ulkoiset tuotteet tai ideat ovat parempia. NIH-ilmiön syynä pidetään vakiintumista tuttuihin menetelmiin ja tiimien kokemaa ylivertaisuutta muiden osaamiseen verrattuna, mikä ilmenee haluttomuutena harkita vakavasti muiden tuottamaa uutta ja innovatiivista informaatiota [19].
Optimismiharha	Systemaattista optimististen odotusten suosimista kutsutaan *optimismiharhaksi*. Tällöin ihmisillä on taipumus yliarvioida positiivisten tapahtumien todennäköisyyttä ja vastaavasti uskoa negatiivisten tapahtumien todennäköisyyden olevan todellista matalampi [13]. Optimistisuus ja pessimistisyys ovat eroja ihmisen odotusten ja todellisen lopputuloksen välillä. Jos yksilön odotukset ovat lopputulosta positiivisempia, ne ovat optimistisesti vääristyneitä.
Edustavuusharha	*Edustavuusharha* on asioiden ja tapahtumien samankaltaisuuden arviointia ja niiden tyypittämistä kategorioihin [3]. Kategoriaan kuuluvilla esineillä ja tapahtumilla uskotaan tällöin olevan samoja ominaisuuksia ja seurauksia. Edustavuusharhan takia ihmiset olettavat esineillä tai asioilla olevan kaikki samat ominaisuudet mitä muillakin kategoriaan kuuluvilla asioilla.

Valintaharha	*Valintaharha* saa ihmiset kiinnittämään joihinkin vaihtoehtoihin enemmän huomiota kuin toisiin, koska niillä on jokin mieluisa ominaisuus [5]. Tämän takia kokonaiskuvan kannalta paras ratkaisu saattaa jäädä valitsematta, koska muita vaihtoehtoja, joilta tämä ominaisuus puuttuu, ei edes harkita vakavasti.

Taulukko 1: Kognitiivisia vinoumia.

2.4 Vinoumien vaikutus ohjelmistotuotannossa

Koska kaikilla ihmisillä esiintyy kognitiivisia vinoumia, niitä voi ilmetä myös ohjelmistokehittäjillä. Ohjelmistojen suunnittelu perustuu pitkälle ihmisen ajatteluprosesseihin: ohjelmistokehittäjiltä vaaditaan päätöksentekokykyä sekä päättely- ja ongelmanratkaisutaitoja. Näitä ajatteluprosesseja tarvitaan ohjelmistojen suunnittelussa, kehittämisessä ja testauksessa.

Ohjelmistojen suunnittelu on monimutkaista ja ne sisältävät valtavasti erilaisia vaatimuksia ja teknistä informaatiota [1]. Kaikkea informaatiota on mahdotonta sisäistää ja päätöksiä saatetaan joutua tekemään puutteellisilla tiedoilla. Usein kokeneiden ohjelmistokehittäjien intuitiivisesti tehdyt päätökset ovat kuitenkin hyviä, sillä pitkäkestoisen alan harjoittamisen takia kehittäjät pystyvät palauttamaan mieleen ongelmiin sopivia ratkaisuja [10]. Kun osaamiseen ja tietoon perustuvan sopivan ratkaisun tuottaminen epäonnistuu, päätöksenteko voi johtaa päättelyvirheisiin.

Virheet päättelyssä voivat johtaa huonoihin valintoihin ja päätöksiin, jotka voivat heikentää ohjelmiston laatua tai pahimmassa tapauksessa koko projekti voi epäonnistua [4]. Kognitiiviset vinoumat ovat todennäköisesti yksi syy ohjelmistotuotannossa tehtäviin huonoihin päätöksiin [20].

Vinoumilla voi olla monenlaisia vaikutuksia ohjelmistosuunnittelussa. Niitä voi esiintyä vaatimusmäärittelyssä, suunnittelussa, testauksessa ja projektinhallinnassa [14]. Esimerkiksi komplekseja ongelmia saatetaan yksinkertaistaa, jotta niiden ratkaiseminen olisi helpompaa [4]. Koska koko ongelma-alueen täydellinen tutkiminen ei ole mahdollista, saatetaan tyytyä ensimmäiseen ratkaisuun, joka ratkaisee ongelman riittävän hyvin. Useita vaihtoehtoja ei mahdollisesti harkita ollenkaan.

Koska ohjelmistokehittäjät haluavat todentaa, että heidän toteuttamansa ohjelmisto toimii, he saattavat testata ohjelmaa tavalla, joka vahvistaa ohjelman toimivuuden sen sijaan että siitä yritetään etsiä virheitä [14]. Kehittäjät saattavat myös etsiä vain sellaista informaatiota, joka vahvistaa heidän uskomuksiaan ja perustella muille ja itselleen valintojensa paremmuuden keskittymällä ainoastaan valintansa vahvuuksiin. Omien päätelmien, hypoteesien ja uskomusten virheellisyyttä ei mahdollisesti pohdita ollenkaan [4].

Monia ohjelmistotuotannossa esiintyviä yleisimpiä ongelmia voidaan selittää kognitiivisilla vinoumilla [14]. Koska ohjelmistokehittäjät pyrkivät vahvistamaan että

koodi toimii, ohjelmistoon voi huomaamatta jäädä helposti virheitä, mikä heikentää ohjelmiston laatua ja aiheuttaa kustannuksia. Työmääräarvioiden on tapana olla usein liian optimistisia, jonka seurauksena ohjelmistoprojektien aikataulut voivat myöhästyä ja budjetit ylittyä.

2.5 Vinoumien vaikutuksen vähentäminen

Koska kognitiiviset vinoumat voivat vaikuttaa haitallisesti päätöksentekoon, menetelmien löytäminen vinoumien poistamiseen tai niiden vaikutuksen vähentämiseen on perusteltua. Osa ohjelmistoprojekteissa esiintyvistä ongelmista johtuu todennäköisesti vinoumista, joten keinoja vinoumien heikentämiseksi olisi mielenkiintoista tutkia myös ohjelmistotuotannon näkökulmasta. Vähentämiskeinojen kehittäminen voi kuitenkin osoittautua haasteelliseksi, koska moni tutkituista keinoista vinoumien poistamiseen on osoittautunut tehottomaksi, sillä kognitiiviset vinoumat ovat hyvin sisäänrakennettu osa ihmisen ajattelua.

Menetelmät vinoumien vähentämiseen voidaan jakaa kolmeen kategoriaan sen mukaan, mistä syystä vinoumien oletetaan johtuvan. Syitä voivat olla joko vialliset tehtävät, vialliset päätöksentekijät tai tehtävän ja päätöksentekijän välinen sopimattomuus [21].

Tehtävä vinoumien tunnistamisesta ja torjumisesta voidaan asettaa ensinnäkin päätöksentekijän vastuulle. Tutkimusten mukaan on kuitenkin havaittu, että pelkästään vinoumien olemassaolosta kertominen ei vähennä vinoumien vaikutusta [11]. Vinoumia ei voi myöskään vähentää kehottamalla ihmisiä yrittämään kovemmin tai lisäämällä heidän päätöstensä vastuullisuutta [21].

Oikeista päätöksistä palkitseminenkaan ei ole välttämättä tehokas keino: tällöin päätöksenteossa onnistuminen vaatisi, että yksilöllä on kaikki vaadittavat taidot päätöksentekoon, ja ainoa rationaalista päätöksentekoa haittaava asia olisi yrityksen puute. Useimmissa tapauksissa siis myöskään palkitseminen ei toimi [22].

Erilaisilla koulutusmenetelmillä on kuitenkin löydetty positiivisia vaikutuksia. Vinoumien tunnistamista opettavalla videopelillä on onnistuttu opettamaan koehenkilöille taitoja, jotka vähentävät vinoumia pitkäkestoisesti [23]. Koulutus ei kuitenkaan takaa, että ihmiset osaavat siirtää laboratorio-olosuhteissa oppimansa taidot todelliseen elämään [22]. Päätöksentekijöiden tulisi osata tunnistaa tilanteet, joissa vinoumia esiintyy, ja heidän pitäisi myös ymmärtää käyttää oppimiaan strategioita. Vinoumien vähentäminen vaatii, että opittu strategia osataan siirtää koulutustilanteesta todelliseen ympäristöön. Koulutuksen haaste onkin siinä, miten ihmisille voidaan opettaa abstrakteja sääntöjä, jotka toimisivat monenlaisissa eri tilanteissa.

Vaikka koulutuksella onkin havaittu positiivisia vaikutuksia, ihmisten on silti hyvin haastavaa muuttaa omaa käyttäytymistään niin, että omaa vinoutunutta ajatteluaan pystyisi vastustamaan. Muiden vinoumia on toki mahdollista havaita ja sitä kautta parantaa muiden päätöksentekoa [11].

Koulutuksen ohella ainut toinen tehokkaaksi osoittautunut tapa vinoumien vähen-

tämiseen on päätöksentekoympäristön muuttaminen niin, ettei vinoumia pääse edes syntymään. Esimerkiksi päätöksentekotilanteen restrukturointi muotoon, joka tekee tehtävästä yhteensopivan päätöksentekijän luonnollisten ajatusprosessien kanssa, on havaittu tehokkaaksi keinoksi ehkäistä vinoumia [21]. Tällöin oletetaan, että päätöksentekijällä on kaikki vaadittavat taidot, mutta jostain syystä näitä taitoja ei käytetä. Vinoumien vähentämisessä onnistutaan, kun tehtävästä ja ihmisen ajatusprosesseista tehdään mahdollisimman hyvin yhteensopivia. Vinoumien vähentämistä tukeva keino on siis restrukturoida päätöksenteko sellaiseen muotoon, joka mahdollistaa päätöksentekijän käyttää kognitiivisia kykyjään tilanteeseen sopivalla tavalla [21].

Päätöksenteon restrukturoinnin keinoja ovat esimerkiksi pyytää päätöksentekijöitä tuomaan kaikki tehtävästä tietämänsä informaatio ilmi eksplisiittisesti ja etsimään ratkaisuista epäjohdonmukaisuuksia vahvistavien perusteluiden sijaan [21]. Myös tehtävän jakaminen pienempiin osiin ja tehtävän muotoilu vaihtoehtoisilla tavoilla mainitaan keinoiksi.

Keinoja, joilla jokainen voi vastustaa vinoumien vaikutusta, on esitetty muutamia. Ne tosin edellyttävät, että päätöksentekijä tunnistaa tilanteet, joissa vinoumia tyypillisesti esiintyy. Esimerkiksi itsevarmuusharhaa ja ankkurointivaikutusta on pystytty onnistuneesti vähentämään pyytämällä päätöksentekijää listaamaan syitä, miksi vastaus tai arvio voisi olla väärin [21, 24]. Kun päätöksentekijä tällä tavoin pakotetaan etsimään vastauksestaan heikkouksia, tämä alkaa kyseenalaistaa päätelmiään ja itsevarmuus vastauksen oikeellisuuteen vähenee.

Kun päätöksentekijää pyydetään antamaan jokin numeerinen arvio, ankkurointivaikutusta voidaan vähentää, kun häntä pyydetään tekemään luvusta kaksi eri arviota. Niiden keskiarvo on todennäköisemmin lähempänä totuutta kuin kumpikaan luvuista [22].

Vinoumien vähentämismenetelmistä voisi olla hyötyä myös ohjelmistoprojektien laadun ja onnistumisen kannalta. Ohjelmistotuotannon menetelmiin olisi täten hyödyllistä sisällyttää vinoumien vähentämistä edistäviä piirteitä.

3 Menetelmä

Tutkielma suoritettiin kirjallisuuskatsauksena. Aineistona käytettiin julkaisuja, joissa tutkittiin kognitiivisia vinoumia ohjelmistotuotannon näkökulmasta. Luvussa määritellään tutkimuskysymykset ja tutkimusstrategia sekä kerrotaan, miten aineisto kerättiin ja analysoitiin.

3.1 Tutkimuskysymykset

Tutkielman tavoitteena oli kartoittaa kognitiivisten vinoumien tutkimusta ohjelmistotuotannon näkökulmasta. Tavoitteena oli selvittää, miten vinoumat voivat ilmetä ohjelmistokehittäjillä ja mitä seurauksia niillä voi olla ohjelmistotuotannossa. Lisäksi haluttiin kartoittaa keinoja, joilla ohjelmistokehittäjien vinoumia voitaisiin vähentää tai estää. Aineistoa analysoimalla haluttiin selvittää, millaisin tutkimusmenetelmin vinoumien tutkimusta on tehty ja miten hyvin ne ottavat huomioon alkuperäiset psykologian tutkimukset kognitiivisista vinoumista.

Tavoitteen saavuttamiseksi muodostettiin seuraavat tutkimuskysymykset:

1. Millaisin tutkimusmenetelmin kognitiivisia vinoumia on tutkittu ohjelmistotuotannon näkökulmasta?
2. Miten kognitiiviset vinoumat voivat vaikuttaa päätöksentekoon ohjelmistotuotannon eri vaiheissa?
3. Miten alkuperäiset psykologian tutkimukset kognitiivisista vinoumista on otettu huomioon?
4. Millaisin keinoin kognitiivisten vinoumien vaikutusta voidaan vähentää ohjelmistotuotannossa?

3.2 Tutkimusstrategia

Aineiston haku kohdistettiin seuraavaan neljään tietokantaan:

- IEEE Xplore Digital Library[1]
- ACM Digital Library[2]
- SpringerLink[3]
- ScienceDirect[4]

[1] `https://ieeexplore.ieee.org/`
[2] `https://dl.acm.org/`
[3] `https://link.springer.com/`
[4] `https://www.sciencedirect.com/`

Haun jälkeen artikkeleita etsittiin lisää snowball-tekniikalla, jossa artikkeleita etsitään lisää sopivan tai sopivien artikkeleiden lähdeviitteistä ja sitaateista. Haku tehtiin molempiin suuntiin, eli etsittiin hakutulosten julkaisuista relevantteja viitteitä, sekä etsittiin julkaisuja, jotka ovat viitanneet hakutuloksien julkaisuihin. Snowball-tekniikkaa käytettiin niin monta kierrosta, kunnes uusia relevantteja julkaisuja ei enää löytynyt.

Aineistoa haettiin seuraavilla hakusanoilla. Haut suoritettiin erikseen, eli kuhunkin tietokantaan tehtiin kuusi eri hakua.

- cognitive bias AND software
- bias AND software
- cognitive bias AND mitigation
- cognitive bias AND prevention
- debiasing AND software
- de-biasing AND software

3.3 Aineiston haku

Hakusanat kohdistettiin julkaisun otsikkoon, abstraktiin sekä avainsanoihin. Aineistoon sisällytettiin vertaisarvioidut konferenssijulkaisut ja tieteelliset lehdet.

Tietokantahaun tuloksista pyrittiin rajaamaan pois ohjelmistotuotantoon liittymättömät tulokset käyttämällä mahdollisuuksien mukaan hakukriteerejä (taulukko 2).

Haku suoritettiin toukokuussa 2019.

Tietokanta	**Hakukriteerit**
IEEE Xplore	Haku kohdistettiin metadataan Publisher: IEEE Sisältötyypit: Conferences, Journals, Magazines
ACM	Publishers: ACM ACM publications: Proceeding, Journal, Magazine Content formats: pdf
SpringerLink	Content type: article, conference paper Disclipine: Computer Science Subdisclipine: Software engineering Language: english
ScienceDirect	Article type: review articles, research articles Publication title: Computers in Human Behaviour (journal)

Taulukko 2: Tietokantahakujen hakukriteerit.

Taulukossa 3 on käytetyt hakusanat sekä yhteenveto tietokantahakujen tulosten määristä tietokannoittain suodatuksen jälkeen. Haku suoritettiin kunkin hakusanan osalta erikseen.

Hakusana	**IEEE**	**ACM**	**Springer**	**Science-Direct**	**Yht.**
cognitive bias AND software	56	242	1033	745	**2076**
bias AND software	1838	242	6715	0	**8795**
cognitive bias AND mitigation	17	92	40	196	**345**
cognitive bias AND prevention	3	36	50	0	**89**
debiasing AND software	2	1	5	4	**12**
de-biasing AND software	3	6295	4	487	**6789**
Yhteensä	**1919**	**6908**	**7847**	**1432**	**18 106**

Taulukko 3: Yhteenveto tietokantahakujen tuloksista.

Tietokantahaun tuloksista valittiin lopulliseen aineistoon 20 julkaisua. Niiden kriteerinä oli, että niissä käsiteltiin jotakin kognitiivista vinoumaa ja että vinouman vaikutuksia tai niiden vähentämiskeinoja tarkasteltiin ohjelmistotuotannon näkökulmasta. Tämä arvioitiin julkaisun otsikon, abstraktin ja loppupäätelmän perusteella. Snowball-tekniikalla saatiin 10 julkaisua lisää. Lopulliseen aineistoon valittiin siis yhteensä 30 julkaisua.

3.4 Aineiston analyysi

Aineistosta poimittiin materiaalia, joka vastasi tutkimuskysymyksiin. Tutkimuskysymyksissä mainittuja aiheita olivat kognitiivisten vinoumien tutkimus ja niiden vaikutus päätöksentekoon ohjelmistotuotannon näkökulmasta sekä keinot vinoumien haitallisten vaikutusten vähentämiseen. Aineiston oli liityttävä siis sekä kognitiivisiin vinoumiin että ohjelmistotuotantoon. Vinoumien haitallisten vaikutusten vähentämistä koskevan materiaalin ei tarvinnut liittyä ohjelmistotuotantoon, sillä materiaalista pyrittiin löytämään ideoita ohjelmistotuotannon alalle tutkielman pohdintaluvussa. Aineistoa löydettiin kuitenkin riittävästi ohjelmistotuotannon näkökulmasta.

Tutkimuskysymyksiin 2 ja 4 (vinoumien vaikutus ja vähentäminen) poimittiin aineistoa julkaisujen abstraktista, tuloksista ja pohdintaluvusta.

Kerättyä aineistoa analysoimalla selvitettiin, miten kognitiivisia vinoumia on tutkittu ohjelmistotuotannon alan näkökulmasta. Analyysissa kartoitettiin, millaisin tutkimusmenetelmin aihetta on tutkittu, eli olivatko tutkimukset kyselyitä, haastatteluita, kokeita vai tapaustutkimuksia.

Aineistoa analysoitiin myös sen kannalta miten alkuperäinen psykologian tutkimus oli otettu julkaisuissa huomioon. Analyysissä selvitettiin, viitattiinko julkaisuissa alkuperäisiin psykologian lähteisiin vinouman määrittelyssä, selitettiinkö vinouman taustamekanismia ja miten psykologian teoriat otettiin huomioon.

Taulukossa 4 on esimerkki siitä, mitä tietoja kustakin artikkelista poimittiin. Artikkeleista poimittiin perustietojen (otsikko, vuosi, julkaisufoorumi) lisäksi tutkittu vinouma, tutkimusmenetelmä, ohjelmistotuotannon vaihe sekä tutkimuksen tausta ja vinoumien havaitut vaikutukset.

Otsikko	A cognitive perspective on pair programming
Vuosi	2006
Julkaisufoorumi	Connecting the Americas. 12th Americas Conference on Information Systems, AMCIS
Vinouma	Optimismiharha, ankkurointivaikutus
Tutkimusmenetelmä	Kontrolloitu koe
Ohjelmistotuotannon vaihe	Suunnittelu
Tausta	In this research, we focus on empirically investigating the impact of pairing experts and novices in different ways on their productivity. When working together on a problem solving task, developers may exhibit different levels of cognitive biases as a result of their expertise and this in turn may affect the performance of the pair. Our research objective is to examine the relationship between pair composition, cognitive biases, and productivity.
Havaitut seuraukset	1. Experts exhibit confidence biases as they were reluctant to examine documentation. Novices were less susceptible to this bias. 2. Experts were more willing to explore wide range of solutions rather than anchoring to just one solution. Novices, due to their inability to identify such wider range of solutions, tended to anchor to their initial solution. 3. Also, propensity of experts to self-evaluate led them to make recurring adjustments to their solutions, unlike the novices who rarely evaluated quality of their solutions, resulting in little or no adjustments to their solutions.

Taulukko 4: Esimerkki poimituista tiedoista yhdestä artikkelista.

4 Tulokset

Luvussa esitetään tutkimuksen tulokset kerätyn aineiston pohjalta. Aineisto on jaettu kahteen osaan: vinoumien tutkimukseen ja niiden vaikutuksen vähentämistä käsittelevään aineistoon. Suuri osa aineiston artikkeleista kuitenkin käsitteli aihetta molemmista näkökulmista. Tulokset kerrotaan kustakin näkökulmasta erikseen.

Tuloksissa selostetaan mitä vinoumia aineistoon liittyvissä julkaisuissa tutkittiin ja mitä tutkimusmenetelmiä tutkimuksissa käytettiin. Lisäksi selvitettiin, missä ohjelmistotuotannon vaiheissa vinoumia tutkittiin.

Psykologian tutkimuksen hyödyntämisestä kerrotaan, kuinka hyvin alkuperäisiin psykologian lähteisiin viitattiin, selitettiinkö vinoumien taustamekanismia ja otettiinko psykologian teoriat huomioon.

Ohjelmistotuotannossa havaitut vinoumat esitetään kunkin vinouman osalta erikseen. Myös aineistosta löytyneet keinot vinoumien poistamiseen tai vähentämiseen selostetaan vinouma kerrallaan.

4.1 Tutkitut vinoumat

Aineistossa käsiteltiin kymmentä erilaista kognitiivista vinoumaa, jotka on listattu taulukossa 5. Aineisto on jaettu kahteen osaan: vinoumien tutkimukseen ja niiden vaikutusten vähentämiseen ohjelmistotuotannossa. Julkaisujen määrä on esitetty taulukossa kummastakin näkökulmasta erikseen.

Eniten tutkitut vinoumat olivat ankkurointivaikutus ja vahvistusharha. Muita vinoumia oli tutkittu yhdessä tai kahdessa artikkelissa.

Monet julkaisut käsittelivät vinouman tutkimuksen lisäksi keinoja sen vähentämiseksi. Ankkurointivaikutusta ja vahvistusharhaa oli tutkittu eniten vähentämisen näkökulmasta, optimismiharhaa kolmanneksi eniten. Ikea-efektin vähentämistä ei ollut tutkittu lainkaan.

Seuraavat aineistosta löytyneet vinoumat niputettiin saman termin alle: *optimism bias* ja *over-optimism* optimismiharhan alle, sekä *confirmation bias* ja *confirmatory bias* vahvistusharhan alle. Myös *positive test bias* on laitettu vahvistusharhan alle, koska kyse on pohjimmiltaan samasta ilmiöstä.

4.2 Tutkimusmenetelmät

Vinoumien tutkimuksessa selvästi suosituin tutkimusmenetelmä oli kokeellinen tutkimus. Suurin osa suoritetuista kokeista oli kontrolloituja; muut kolme olivat kvasikokeita. Aineistossa oli myös yksi lehtiartikkeli, katselmointi ja kysely. Taulukossa 6 on yhteenveto aineiston julkaisuissa käytetyistä tutkimusmenetelmistä.

Vinoumien vähentämistä koskevassa aineistossa kokeellista tutkimusmenetelmää (kontrolloitu tai kvasikoe) käytettiin ainoastaan seitsemässä julkaisussa. Aineiston yhteen-

veto on taulukossa 7. Selvästi suurin osa vinoumien vähentämiskeinoista oli ainoastaan ehdotuksia tai hypoteeseja, eli niiden pätevyydestä ei ollut tieteellistä näyttöä.

Vinouma	**Vinoumia koskeva aineisto**	**Vinoumien vähentämistä koskeva aineisto**
Ankkurointivaikutus	[A1] [A2] [A3] [A4] [A5], [A6] [A7]	[A4] [A5] [A7] [A8] [A9] [A10] [A11]
Saatavuusharha	[A12] [A13]	[A10] [A12] [A13]
Vahvistusharha	[A12] [A14] [A15] [A16] [A17] [A18]	[A10] [A12] [A14] [A15] [A16] [A17] [A18]
Kehysvaikutus	[A19]	[A10], [A20], [A19]
Ikea-efekti	[A21]	
Not-invented-here	[A22]	[A22]
Optimismiharha	[A23]	[A10], [A23], [A24], [A25], [A26]
Itsevarmuusharha	[A7]	[A7], [A10], [A27]
Edustavuusharha	[A12], [A28]	[A10], [A12], [A28]
Valintaharha	[A29]	[A29]

Taulukko 5: Tutkitut vinoumat.

Tutkimusmenetelmä	**Vinoumia koskeva aineisto**
Kontrolloitu koe	[A1] [A3] [A4] [A5] [A6] [A7] [A14] [A15] [A28] [A19] [A21] [A23] [A17] [A30] [A16]
Kvasikoe	[A2] [A13] [A29]
Lehtiartikkeli	[A12]
Katselmointi	[A18]
Kysely	[A22]

Taulukko 6: Käytetyt tutkimusmenetelmät vinoumien tutkimusta koskevassa aineistossa.

Tutkimusmenetelmä	**Vinoumien vähentämistä koskeva aineisto**
Kontrolloitu koe	[A4] [A8] [A9] [A14] [A24] [A25]
Kvasikoe	[A26]
Ehdotus tai hypoteesi	[A10] [A29] [A5] [A7] [A11] [A12] [A13] [A15] [A16] [A17] [A18] [A19] [A20] [A22] [A23] [A27] [A28]

Taulukko 7: Käytetyt tutkimusmenetelmät vinoumien vähentämistä koskevassa aineistossa.

4.3 Ohjelmistotuotannon vaiheet

Vinoumia on tutkittu monipuolisesti eri ohjelmistotuotannon vaiheiden näkökulmasta. Tutkimusta on tehty vaatimusanalyysissä, suunnittelussa ja testauksessa esiintyvistä vinoumista. Spesifisemmistä ohjelmistoprojektien aktiviteeteista vinoumien tutkimusta löytyi työmääräarvioinnin, projektinhallinnan sekä tarjouskilpailun näkökulmista. Tulosten yhteenveto on taulukossa 8.

Vinoumia on tutkittu eniten testauksessa, suunnittelussa ja työmääräarvioinnissa. Muutamia yksittäisiä tutkimuksia on tehty vaatimusanalyysin, projektinhallinnan ja tarjouskilpailuiden näkökulmasta. Julkaisuja löytyi kutakuinkin yhtä paljon vinoumia ja niiden vähentämistä koskevissa aineistoissa.

Vaihe tai aktiviteetti	**Vinoumia koskeva aineisto**	**Vinoumien vähentämistä koskeva aineisto**
Vaatimusanalyysi	[A19] [A21]	[A9] [A20] [A19]
Suunnittelu	[A1] [A3] [A6] [A7] [A12] [A22]	[A7] [A9] [A10] [A11] [A22]
Testaus	[A14] [A15] [A16] [A17] [A18]	[A10] [A12] [A14] [A15] [A17] [A18]
Työmääräarviointi	[A2] [A4] [A5] [A7] [A28] [A30]	[A4] [A5] [A8] [A10] [A24] [A25] [A26] [A27] [A28]
Projektinhallinta	[A13] [A23]	[A10] [A13] [A23]
Tarjouskilpailu	[A29]	

Taulukko 8: Tutkitut vinoumat ohjelmistoprojektien vaiheissa ja aktiviteeteissa.

4.4 Psykologian tutkimuksen hyödyntäminen

Seuraavaksi kerrotaan, miten aineistossa otettiin huomioon psykologian tutkimus ja teoriat. Tätä haluttiin selvittää siksi, että ohjelmistotuotannon kognitiivisten vinoumien tutkimuksen laatuun vaikuttaa se, kuinka hyvin psykologian tutkimusta on osattu niissä hyödyntää. Julkaisuista selvitettiin, viitattiinko julkaisuissa alkuperäisiin psykologian lähteisiin, selitettiinkö vinouman taustamekanismia ja otettiinko psykologian teoriat kognitiivisista vinoumista huomioon.

4.4.1 Viittaukset alkuperäisiin psykologian lähteisiin

Useimmissa tutkimuksissa viitattiin alkuperäiseen psykologian tutkimuksen lähteeseen. Tulokset on esitetty taulukossa 9.

Yhdessä tutkimuksessa viittausta ei ollut tutkimuksen alla olevaan vinoumaan, vaan siinä määriteltiin kognitiivinen vinouma ainoastaan yleisellä tasolla [A1].

Kolmessa tutkimuksessa vinouman määrittelyssä ei ollut viittausta lainkaan [A4, A17, A29].

Viittaus	Lähteet
Kyllä	[A2] [A3] [A5] [A6] [A7] [A12] [A13] [A14] [A15] [A16] [A18] [A19] [A21] [A22] [A23] [A28] [A30]
Osittain	[A1]
Ei	[A29] [A4] [A17]

Taulukko 9: Viittaukset alkuperäisiin psykologian lähteisiin.

4.4.2 Vinouman taustamekanismin selittäminen

Vinoumien taustamekanismia selitettiin vain seitsemässä julkaisussa. Tulokset ovat kuvattu taulukossa 10.

Edustavuusvaikutuksen ja saatavuusharhan kerrottiin johtuvan aiempien kokemusten mentaalisista representaatioista [A12]. Tyypillisten kokemusten tai asioiden representaatiot saavat meidät kiinnittämään huomiota näihin ominaisuuksiin helpommin, jolloin ne palautuvat mieleen nopeammin kuin epätavalliset kokemukset tai asiat.

Vinoumien taustamekanismiksi mainittiin ongelmanratkaisun yksinkertaistaminen ja kognitiivisen kuorman minimointi [A1, A3]. Mielen pyrkimystä yksinkertaistaa tehtävät hallittavimpiin osuuksiin pidettiin myös syynä vinoumien olemassaoloon [A7].

Heuristiikat eli yksinkertaiset päätöksentekosäännöt mainittiin vinoumien aiheuttajiksi kahdessa julkaisussa [A4, A28].

Ankkurointivaikutuksen kerrottiin johtuvan semanttisen tiedon aktivoitumisesta ja taipumuksesta harjoittaa hypoteesin vahvistavaa testausta [A5].

Selitys	Lähteet
Kyllä	[A1] [A4] [A3] [A5] [A7] [A12] [A28]
Ei	[A2] [A6] [A13] [A14] [A16] [A17] [A18] [A19] [A21] [A22] [A23] [A29]

Taulukko 10: Vinouman taustamekanismin selittäminen.

4.4.3 Psykologian teoriat

Aineistosta tutkittiin, miten hyvin julkaisuissa käsiteltiin psykologian teorioita kognitiivisista vinoumista.

Selvästi suurin osa julkaisuista otti psykologian teoriat huomioon. Niissä selostettiin vinouman tutkimukseen liittyviä tutkimustuloksia. Ainoastaan kolme julkaisua ei maininnut psykologian teorioista tai tutkimustuloksista lainkaan [A13, A22, A29]. Tulokset on esitetty taulukossa 11.

Kolmessa julkaisuissa tutkimuksen koe perustui aiempiin psykologian tutkimuksen

Viittaus	Lähteet
Kyllä	[A1] [A2] [A3] [A4] [A5] [A6] [A7] [A12] [A14] [A16] [A17] [A18] [A19] [A21] [A23] [A28] [A30]
Ei	[A13] [A22] [A29]

Taulukko 11: Psykologian teorioiden huomioon ottaminen.

kokeisiin [A14, A15, A16].

Neljässä julkaisuissa selostettiin vinouman tutkimustuloksia ja vaikutuksia ja pääteltiin sen perusteella, että ne ilmenevät myös ohjelmistotuotannossa [A5, A12, A19, A21]. Muodostettua hypoteesia ei kuitenkaan vahvistettu millään tavalla.

Kahdessa julkaisussa tutkittiin, onko vinoumalla sama vaikutus myös ohjelmistotuotannossa kuin psykologian tutkimuksen kokeissa [A2, A4].

4.5 Vinoumien vaikutus ohjelmistotuotannossa

Luvussa kerrotaan, mitä vinoumia ohjelmistotuotannon alalla on havaittu, miten ne voivat ilmetä, ja mitä seurauksia niillä voi olla ohjelmistotuotannossa.

Taulukossa 12 on yhteenveto vinoumien vaikutuksista ja niitä voimistavista tekijöistä ohjelmistotuotannossa. Vaikutukset käydään kunkin vinouman osalta tarkemmin läpi aliluvuissa.

Vinouma	Vaikutus
Ankkurointivaikutus	SQL-kyselyihin ankkuroituminen niiden uudelleenkäytössä [A1] Ankkuroituminen mihin tahansa lukuun työmääräarvioinnissa [A2, A30] Työmääräarviointi tehtäviä pilkkomalla [A2] Työmääräarviointi ilman menetelmää [A2] Aikavälien käyttäminen työmääräarvioinnissa [A30] Toteutettuun ratkaisuun ankkuroituminen koodia uudelleenkäytettäessä; ylimääräisten ominaisuuksien siirtyminen uuteen kohteeseen [A3] Vähäinen kokemus ohjelmistosuunnittelusta [A7]
Edustavuusharha	Mieleenpainuvampien koodin osien ylirepresentaatio [A12] Määrittelydokumentaatiossa olevan ylimääräisen informaation vaikutus työmääräarvioihin [A28]
Ikea-efekti	Ylimääräisten ominaisuuksien määrittely ja haluttomuus luopua niistä [A21] Ominaisuuksien pitäminen alkuperäistä tärkeämpänä kun sen toteuttamiseen käytetään aikaa [A21]

Itsevarmuusharha	Kokemus ohjelmistosuunnittelusta; haluttomuus tutkia dokumentaatiota [A7] Liiallinen itsevarmuus työmääräarvioiden paikkansapitävyydestä [A2]
Not-invented-here	Yrityksen ulkopuolelta tulevien ideoiden ja komponenttien hyötyjen aliarviointi [A22]
Optimismiharha	Optimistiset työmääräarviot [A2] Hallinnan tunne riskeistä, kun projektin riskit ymmärretään hyvin tai kun riskit on kuvattu korkealla tasolla [A23]
Kehysvaikutus	Määriteltyjen ominaisuuksien lopullisuutena pidetty taso kun niitä kutsutaan vaatimuksiksi tai ideoiksi [A19] Kysymyksen muotoilu työmääräarvoinnissa; kuinka paljon saadaan tehtyä tietyssä ajassa [A5]
Saatavuusharha	Helposti muistettavan koodin ja ominaisuuksien tarkempi käsittely [A12] Aiempi tietämys dokumentaatiosta ohjaa hakua [A13] Avainsanoilla haku dokumentaatiossa; tutut ja helpommin muistettavat avainsanat ohjaavat hakua [A13]
Vahvistusharha	Aiempi tietämys dokumentaatiosta ohjaa hakua [A13] Positiivisten testitapausten suosiminen [A17, A18] Vähäinen kokemus testaamisesta [A17] Ohjelmointivirheiden etsinnässä yhden hypoteesin testaaminen kerrallaan [A12] Hypoteesin testaustaitojen ja loogisen päättelyn harjoittamisen puute [A14]
Valintaharha	Teknologioiden, arkkitehtuurien tai kehitystyökalujen valinta kustannusten tai työmäärän perusteella [A29] Hinnan korostaminen tarjouskilpailussa [A29]

Taulukko 12: Vinoumien vaikutuksia ja niitä voimistavia tekijöitä ohjelmistotuotannossa.

4.5.1 Ankkurointivaikutus

Ankkurointivaikutusta on havaittu esiintyvän SQL-kyselyiden muodostamisessa [A1], työmääräarvioinnissa [A2, A30] ja uudelleenkäytettäessä koodia [A3].

SQL-kyselyiden muodostaminen on nopeampaa, kun kyselyitä käytetään uudelleen, mutta se voi johtaa virheellisiin tai heikompilaatuisiin kyselyn tuloksiin ja korkeampaan itsevarmuuteen tulosten oikeellisuudesta. Kun kyselyitä ei kirjoiteta tyhjästä, vaan ne muodostetaan aiemman kyselyn pohjalta, kehittäjät ovat alttiita ankkurointivaikutukselle, jolloin ankkuroidutaan alkuperäiseen kyselyn muotoon [A1]. Tällöin ei välttämättä havaita, että alkuperäinen kysely ei olekaan välttämättä optimaalinen uudessa kontekstissa ja että se voi sisältää jopa virheitä.

Tutkimuksen mukaan kehittäjät ovat myös varmempia siitä, että SQL-kysely on muodostettu oikein, kun kyselyyn katsotaan mallia. Ankkuroituminen on voimakkaampaa silloin, kun muutokset ovat ”syvän tason” rakennemuutoksia, kuten liitoksia, kuin jos muutokset ovat ”pintatason” rakennemuutoksia, kuten projektioita tai ehtolausekkeita. Eli jos kysely vaatii yksinkertaisia muutoksia, ankkuroituminen on heikompaa kuin jos kysely vaatii useita tai monimutkaisia muutoksia, joilla voi olla odottamattomia vaikutuksia tuloksiin [A1, A6].

Ankkuroinnin on osoitettu vaikuttavan ohjelmistoprojektien työmääräarvioihin. Arvioijilla on myös taipumus olla liian itsevarmoja omista arvioistaan [A2]. Työmääräarvioita tehdessä ympäristössä voi olla läsnä erilaisia lukuja tai toisten arvioijien tuottamia arvioita työmääristä. Nämä luvut voivat toimia ankkurina, jotka vaikuttavat siihen, minkä suuruiseen työmääräarvioon arvioija päätyy. Kun arvioijille annetaan korkea ankkuriluku, heidän työmääräarvionsa ovat merkittävästi korkeampia, kuin jos annetaan matala ankkuri tai ei ollenkaan ankkuria [A2]. Tutkimuksessa korkean ankkuriluvun tapauksessa työmääräarviot olivat keskimäärin kaksi kertaa suurempia kuin matalan ankkurin tapauksessa.

Ankkuriluvut voivat vaikuttaa työmääräarvioihin hyvin merkittävästi. Koehenkilöitä pyydettiin antamaan työmääräarvio kahdesta eri skenaariosta: niin että työ sujuu mahdollisimman hyvin (”parhain skenaario”) tai jos kaikki menee mahdollisimman huonosti (”pahin skenaario”). Lisäksi heille annettiin joko korkea tai matala ankkuriluku. Ankkurilla oli niin voimakas vaikutus, että koehenkilöt saatiin antamaan pahimmalle skenaariolle optimistisempia työmääräarvioita kuin parhaalle skenaariolle: matalan ankkurin pohjalta luotu ”pahimman skenaarion” työmääräarvio oli matalampi kuin korkean ankkuriin perustuvan ”parhaan tapauksen” työmääräarvio, mikä on ristiriitaista, sillä pahimman skenaarion arvion tulisi saada korkeampia arvioita kuin parhaimman skenaarion [A2]. Havainto korostaa miten epäluotettavia työmääräarviot voivat olla, kun päätöksentekijät altistuvat ankkuroitavalle luvulle.

Tehtävien pilkkominen tai ohjelmiston koon mittaamiseen perustuvat menetelmät työmääräarvioinnissa vaikuttavat ankkurointivaikutuksen voimakkuuteen. Ankkurointivaikutuksen havaittiin olevan voimakkaampi, kun työmääräarviot tehtiin tehtävien pilkkomiseen perustuvalla menetelmällä tai kun arvioinnissa ei käytetty minkäänlaista menetelmää [A2]. Käytettäessä koodiriveihin (LOC) tai toimintopisteisiin (FP) perustuvia menetelmiä ankkurointivaikutus oli vähäisempi.

Työmääräarvioinnissa ankkurin tarkkuudella ja lähteen luotettavuudella ei ollut vaikutusta vinouman voimakkuuteen [A30]. Kokeessa käytettiin tarkkoja lukuja, pyöristettyjä lukuja ja aikavälejä. Ankkurointivaikutus oli voimakkaampi, kun käytettiin aikavälejä verrattuna yksittäisiin lukuihin [A30]. Ankkuriluku aiheutti ankkuroitumista sekä luotettavien lähteiden tapauksessa ja myös silloin kun koehenkilöt tiesivät ettei ankkurina toimiva luku liity arviointiin. Eli jopa epäluotettavista ja projektiin liittymättömistä lähteistä tulevat luvut voivat vaikuttaa työmääräarvioihin.

Ankkuroitumista voi tapahtua myös uudelleenkäytettäessä koodia tai ohjelmistoartefakteja, kuten arkkitehtuuria, vaatimuksia tai suunnitteludokumentteja. Tällöin artefakteissa olevat virheet saattavat siirtyä uuteen kohteeseen tai niissä voi olla

ylimääräisiä tai puuttuvia ominaisuuksia. Tutkimuksessa löydettiin vahvaa näyttöä siitä, että lopullisessa ratkaisussa käytetään uudelleenkäytettävän artefaktin ylimääräisiä ominaisuuksia [A3]. Sekä kokeneilla että kokemattomilla kehittäjillä esiintyi samaa käyttäytymistä. Koehenkilöt eivät kuitenkaan ankkuroituneet artefaktien virheisiin eivätkä unohtaneet lisätä tarvittavia ominaisuuksia, mutta tietokannan suunnittelutehtävässä kokemattomat kehittäjät (opiskelijat) ankkuroituivat virheisiin sekä puuttuviin ominaisuuksiin [A3].

Ohjelmistokehittäjien kokemus vaikuttaa siihen, miten helposti he ankkuroituvat keksimiinsä ratkaisuihin. Kokeneet ohjelmistokehittäjät ovat valmiita miettimään käsillä olevaan ongelmaan useita ratkaisuja. Sen sijaan kokemattomilla kehittäjillä on taipumus ankkuroitua ensimmäiseen keksimäänsä ratkaisuun [A7]. Kokeneet kehittäjät myös parantelivat ratkaisujaan; kokemattomat harvoin arvioivat ratkaisunsa laatua.

4.5.2 Edustavuusharha

Edustavuusharhan takia kehittäjät saattavat kiinnittää enemmän huomiota niihin koodin osiin, jotka ovat mieleenpainuvampia (kuten esimerkiksi helposti muistettavat muuttujien nimet) ja joiden kanssa he ovat työskennelleet eniten [A12]. Tällöin näitä koodin osia arvioidaan eniten ja muut osat jäävät vähemmälle huomiolle.

Vanhentuneet ja ylimääräiset vaatimukset määrittelydokumentaatiossa voivat johtaa suurempiin työmääräarvioihin. Kun koehenkilöitä pyydettiin olemaan ottamatta huomioon ylimääräisiä vaatimuksia, työmääräarvioista tuli korkeampia kuin jos ylimääräisistä vaatimuksista ei mainittu koehenkilöille lainkaan [A28]. Syyksi arvellaan sitä, että teksti, jossa pyydetään koehenkilöitä sivuuttamaan vaatimuksia, saa koehenkilön luulemaan että kyseessä on suurempi ja monimutkaisempi järjestelmä kuin todellisuudessa.

4.5.3 Ikea-efekti

Ikea-efekti saa kehittäjät kiintymään tekemiinsä ja määrittelemiinsä ohjelmiston ominaisuuksiin [A21]. Ilmiö voi johtaa siihen, että ohjelmistoon määritellään liikaa ominaisuuksia vaatimusten ja asiakkaan tarpeiden ulkopuolelta, koska myöhemmin ylimääräisiksi osoittautuvista ominaisuuksista ei haluta luopua niihin käytetyn ajan ja vaivan takia.

Mitä hankalampi ja aikaa vievämpi tehtävä on suorittaa, sitä enemmän arvoa kehittäjä asettaa toteuttamalleen ominaisuudelle ja sitä voimakkaampi ikea-efekti on [A21]. Vaikka ominaisuus olisi määrittelyvaiheessa toissijainen, mutta sen toteuttamiseen joudutaan käyttämään paljon vaivaa, ominaisuutta aletaan pitää alkuperäistä tärkeämpänä. Tällöin ominaisuutta ei välttämättä haluta tarpeen vaatiessa jättää pois, vaikka ominaisuus ei olisi olennainen ohjelmiston kannalta. Eli kun kehittäjä käyttää aikaa ja vaivaa ominaisuuden määrittelemiseen ja kehittämiseen, hän kiintyy ominaisuuteen emotionaalisesti ja alkaa pitää sitä todellista tärkeämpänä.

4.5.4 Itsevarmuusharha

Kokeneilla kehittäjillä on taipumusta liialliseen itsevarmuuteen, koska he eivät ole yhtä halukkaita tutkimaan dokumentaatiota kuin kokemattomat kehittäjät [A7]. He uskovat tuntevansa järjestelmän niin hyvin, ettei heidän tarvitse tarkistaa käsityksiään dokumentaatiosta.

4.5.5 Not-invented-here

Not invented here -ilmiön takia kehittäjät saattavat olla haluttomia käyttämään muualla toteutettuja ohjelmistokomponentteja [A22]. Kehittäjät saattavat kokea, että muiden luomat komponentit voivat rajoittaa omaa luovuutta. Kehittäjät saattavat antaa enemmän arvoa sisäisesti kehitetyille tai itse tekemilleen komponenteille ja aliarvioida muiden kehittämien komponenttien hyötyjä [A22]. NIH-ilmiö vaikuttaa sekä tiimi- että yksilötasolla.

4.5.6 Optimismiharha

Mitä parempi ymmärrys kehittäjillä on projektin riskeistä, sitä optimistisempi käsitys heillä on niiden hallinnasta [A23]. Sen seurauksena riskinhallintasuunnitelman valinnassa keskitytään enemmän suunnitelman hyötyihin kuin haittoihin. Riskien määritelmän taso vaikuttaa siihen, miten niihin suhtaudutaan: korkealla tasolla kuvattuihin riskeihin suhtaudutaan optimistisemmin kuin konkreettisiin, yksityiskohtaisella tasolla kuvattuihin riskeihin [A23].

4.5.7 Kehysvaikutus

Kun tutkimuksessa ohjelman toivottuja ominaisuuksia kutsuttiin *vaatimuksiksi*, koehenkilöt tuottivat vähemmän luovia ratkaisuja kuin jos niitä kutsuttiin *ideoiksi* [A19]. Ominaisuuksien esittäminen vaatimuksina aiheuttaa siis fiksaatiota, joka rajoittaa luovuutta. Ominaisuuksia pidetään tällöin hyvin ymmärrettyinä ja lopullisina.

Kysymyksen muotoilulla on väliä, kun pyydetään työmääräarvioita. Koehenkilöitä pyydettiin tekemään työmääräarvioita kysymällä perinteisellä tavalla ”Kuinka kauan vaaditaan toteuttamaan x?” (”*How much effort is required to complete X?*”) tai vaihtoehtoisella tavalla ”Kuinka paljon voidaan toteuttaa y tunnissa?” (”*How much can be completed in Y work-hours?*”). Vaihtoehtoisella tavalla arviot olivat optimistisempia samasta työmäärästä [A5].

4.5.8 Saatavuusharha

Saatavuusharhan takia kehittäjät ovat taipuvaisia arvioimaan enemmän helposti muistettavaa koodia ja ominaisuuksia kuin vaikeasti muistettavaa [A12].

Saatavuusharha voi ilmetä ohjelmistojen dokumentaatiota käytettäessä siten, että usein käytetyt dokumentaation osat muistetaan paremmin ja harvoin vieraillut osat heikommin. Aiempi tietämys dokumentaatiosta auttaa kehittäjiä etsimään niistä tietoa tehokkaasti, mutta se voi myös johtaa harhaan ja epätäydelliseen informaation hakuun. Jos dokumentaation rakenne ei vastaa kehittäjän kysymykseen, hän käyttää aiempaa tietämystään dokumentaatiosta tiedon sijainnin ja avainsanojen löytämiseen [A13]. Aiempi tietämys tehostaa etsintää, jos se pitää paikkaansa, mutta jos tietämys dokumentaatiosta on väärää tai vanhentunutta, tietoa etsitään vääristä paikoista dokumentaatiota, jolloin tiedonhaku epäonnistuu tai on puutteellista. Dokumentaation tehoton käyttö voi johtaa turhautumiseen ja siihen, että sen ylläpitoon ei haluta käyttää resursseja. Tehoton käyttö voi johtaa myös siihen, että järjestelmää kehitetään puuttuvien tai väärien tietojen perusteella mikä voi johtaa virheisiin [A13].

Saatavuusharha voi saada kehittäjät sivuuttamaan tiettyjä dokumentaation sijainteja ja avainsanoja, jotka eivät ole tuttuja, vaikka ne voisivatkin johtaa vastaukseen [A13]. Saatavuusharha voi ilmetä myös siten, että avainsanat, joita käytetään usein, muistetaan paremmin ja ovat siten helpommin palautettavissa mieleen, ja siksi niitä aletaan suosia.

4.5.9 Vahvistusharha

Vahvistusharha voi saada kehittäjän etsimään tietoa dokumentaation niistä osista, joiden sijainneista heillä on aiempaa tietämystä. Kokeessa havaittiin, että koehenkilöt eivät käyneet dokumentaatiota läpi kattavasti, vaan he etsivät vastauksen nopeasti joko rakennetta katsomalla tai käyttämällä aiempaa tietämystä ennakoidessaan millä hakusanoilla tai mistä tieto löytyy [A13]. Kehittäjillä voi olla aiempaa tietämystä tai oletuksia tiedon sijainnista, sopivista hakusanoista ja siitä, onko dokumentaatiossa oleva tieto virheetöntä ja lopullista vai vanhentunutta. Väärät olettamukset voivat johtaa dokumentaation tehottomaan käyttöön.

Testauksessa vahvistusharha saa kehittäjän kirjoittamaan positiivisia testejä, jotka vahvistavat hypoteesin, eli että ohjelma toimii vaatimusten mukaan [A17]. Vinouma vaikuttaa siten, että hypoteesia ei pyritä kumoamaan, eli toisin sanoen testauksessa voi jäädä puuttumaan negatiivisia testejä ja löytymättä virheitä. Tutkimuksessa koehenkilöt, jotka tekivät enemmän negatiivisia testejä, löysivät ohjelmasta enemmän virheitä. Toisin sanoen, jos testaajilla esiintyy voimakasta vahvistusharhaa, ohjelmistossa on todennäköisesti enemmän virheitä. Tutkimuksessa koehenkilöt, jotka tuottivat enemmän negatiivisia testejä, eli heillä esiintyi vähemmän vahvistusharhaa, olivat kokeneempia [A17].

Vahvistusharhan takia testaajilla on taipumus tehdä neljä kertaa enemmän positiivisia kuin negatiivisia testejä, jotka vahvistavat, että ohjelma toimii [A18]. Tutkimuksessa positiivisten testien kattavuus oli kaksi kertaa parempi kuin negatiivisten testien. Ilmiö esiintyy kaikilla; ensimmäisen ja useamman vuoden opiskelijoilla, muilla kuin tietojenkäsittelytieteen opiskelijoilla sekä kokeneilla ohjelmoijilla. Kokeneilla

testaajilla vinouma oli heikompi [A18].

Vahvistusharha voi vaikuttaa ohjelmointivirheiden etsintään ja korjaamiseen. Jos kehittäjä uskoo löytäneensä virheen syyn, hän alkaa etsiä lisää todisteita siitä, että hän on oikeassa, eikä pyri kyseenalaistamaan oletuksiaan ja harkitsemaan syyksi useampia vaihtoehtoja [A12]. Kehittäjillä on taipumus testata yhtä hypoteesia kerrallaan, koska se on kognitiivisesti helpompaa kuin useiden hypoteesien muodostaminen ja testaaminen yhtäaikaisesti virheen sijainnin etsinnässä.

Kehittäjien tekemien virheiden määrä on yhteydessä vahvistusharhan voimakkuuteen. Eli mitä voimakkaampi vahvistusharhan taso, sitä enemmän koodissa on virheitä [A14]. Vahvistusharha oli heikompaa niillä osallistujilla, joilla oli hyvät hypoteesin testaustaidot ja jotka olivat harjoittaneet loogista päättelykykyä.

4.5.10 Valintaharha

Valintaharha voi ilmetä kaikissa tilanteissa, joissa valinta tehdään hinnan tai työmäärän perusteella [A29]. Näitä voivat olla esimerkiksi ominaisuuksien valinta ohjelmiston seuraavaan julkaisuun, päätöksestä lähteä kehittämään projektia, tai kun tehdään valintoja teknologioista, arkkitehtuureista tai kehitystyökaluista. Kustannustehokkaimmaksi arvioitu vaihtoehto valitaan todennäköisimmin riippumatta siitä, onko kustannusarvio todenmukainen vai ei [A29]. Valintaharha voi johtaa myös siihen, että tarjouskilpailussa valitaan halvin tarjous, jonka hinta perustuu usein liian optimistiseen työmääräarvioon.

Valintaharha oli voimakkaampi, kun arvioihin liittyi paljon epävarmuuksia ja kun valintatilanteessa korostettiin hinnan tai työmäärän tai molempien merkitystä [A29]. Valintaharhan havaittiin olevan matalampi firman sisäisissä projekteissa ja ohjelmiston ylläpidossa. Mahdollisia syitä tähän voivat olla vaihtoehtojen vähyys, vähäisempi keskittyminen kustannuksiin tai tarkemmat työmääräarviot, jotka johtuvat kontekstin paremmasta ymmärtämisestä.

4.6 Vinoumien vähentäminen ohjelmistotuotannossa

Seuraavaksi kerrotaan, miten kuhunkin vinoumaan voidaan vaikuttaa ja millä keinoin niiden vaikutusta voidaan vähentää ohjelmistotuotannossa. Osa keinoista tutkitusti vähentää vinouman vaikutusta, mutta osa keinoista on ainoastaan ehdotuksia ja oletuksia, eli niiden vaikutustehoa ei ole tutkittu.

Taulukossa 13 on yhteenveto vinoumien vähentämiskeinoista ohjelmistotuotannossa. Vähentämiskeinot käydään kunkin vinouman osalta tarkemmin läpi aliluvuissa.

Vinouma	**Vähentämiskeino**
Ankkurointivaikutus	Jäljitettävyystiedot muutosvaatimusten toteutusta suunniteltaessa [A9] Useiden vaihtoehtojen ja ratkaisujen tuottaminen ongelmaan [A11]; tavan opettaminen opiskeluvaiheessa [A10] Kysymyksen muotoilu työmääräarvioinnissa; keskitytään tehtävään kuluvan ajan sijaan [A5] Aiempien työmäärien hyödyntäminen; selvitetään kuinka kauan vastaavanlaisten tehtävien toteuttamiseen on kulunut [A5] Ankkurointivaikutuksesta kertominen työmääräarvioinnissa [A8] Suunnittelupokeri [A10, A24] Tarkan tai pyöristetyn työmääräarvion suosiminen aikavälien sijaan [A30] Kaikkien lukujen piilottaminen työmääräarvioinnissa [A30] Pariohjelmointi, aivoriihitekniikka [A7]
Edustavuusharha	Ylimääräisen informaation välttäminen dokumentaatiossa [A28]
Itsevarmuusharha	Prosessien ja työkalujen käyttö työmääräarvioinnissa [A27] Pariohjelmointi ja aivoriihitekniikka [A7]
Not-invented-here	Yrityksen kulttuurin kehittäminen; prosessien ja suorituskyvyn mittaaminen ohjelmistoartefakteja valitessa; ulkoisten artefaktien käytöstä palkitseminen [A22]
Optimismiharha	Suunnittelupokeri [A24] Työmääristä kaksi arviota, ihanteellisissa ja realistisissa olosuhteissa [A25] Työmääräarvioista keskusteleminen [A26] Riskien pohtiminen konkreettisella ja yksityiskohtaisella tasolla [A23]
Kehysvaikutus	Ominaisuuksien kehystäminen ideoiksi, kun halutaan luovuutta ja vaatimuksiksi, kun ominaisuuksista ollaan varmoja [A19]
Saatavuusharha	Ontologiaan perustuva dokumentaatio [A13]
Vahvistusharha	Omien päätelmien kyseenalaistaminen ohjelmointivirheiden etsinnässä ja ongelmanratkaisussa [A12] Loogisen päättelykyvyn ja matemaattisten todistustekniikoiden harjoittaminen [A14] Tarkemmat määrittelyt testausta varten; negatiivisten ekvivalenssiluokkien lisääminen määrittelyihin [A17] Vahvistusharhasta ja negatiivisten testien tärkeydestä muistuttaminen [A18] Negatiivisen testin tekeminen jokaista positiivista testiä kohden [A17]

Valintaharha	Arviointiprosessien kehittäminen; historiatietojen käyttö; järjestelmän ymmärryksen parantaminen [A29]

Taulukko 13: Vinoumien vähentämiskeinoja ohjelmistotuotannossa.

4.6.1 Ankkurointivaikutus

Kun ohjelmistokehittäjät saavat tehtäväkseen tehdä ohjelmistoon jonkin muutoksen, he muodostavat mielessään käsityksen muutoksesta ja sen vaatimuksista. Tämä ensimmäinen ymmärrys muutoksesta toimii ankkurina, johon kuuluvat kaikki kehittäjän muodostamat käsitykset ja oletukset siitä. Oletukset voivat olla virheettömiä, jolloin päätökset tehdään oikeiden oletusten pohjalta, mutta ne voivat olla myös virheellisiä, jolloin päätökset tehdään virheellisten oletusten perusteella.

Kehittäjät voivat ankkuroitua muutospyynnön vaatimuksiin, artefakteihin tai ratkaisuihin. Ohjelmistoartefaktien elinkaaren jäljitettävyystietojen avulla ankkuroitumista ensimmäiseen ymmärrykseen voidaan vähentää ja kehittäjät tekevät laadukkaampia muutoksia verrattuna nopeisiin korjauksiin, jotka tehdään ilman tätä tietoa [A9]. Jäljitettävyystiedoista on apua tilanteissa, jotka vaativat monimutkaisia muutoksia, sillä jäljitettävyystiedon ymmärtämiseen kuluu aikaa. Jos kyse on yksikertaisesta muutoksesta, jäljitettävyystietoja ei kannata käyttää apuna, sillä niiden ymmärtäminen saattaa vaatia enemmän kognitiivisia resursseja kuin tehtävän tekeminen.

Ankkuroitumista ensimmäiseen mieleen tulevaan ratkaisuun voitaisiin vähentää sillä, että ohjelmistosuunnittelijoita kehotettaisiin tuottamaan ongelmiin useita vaihtoehtoja ja ratkaisuja, jotta ensimmäinen mieleen tuleva ratkaisu ei alkaisi ohjata ajattelua [A11]. Useiden vaihtoehtoisten ratkaisujen tuottamisesta saattaisi tulla luontaista, jos kehittäjiä pyydettäisiin luomaan niitä jo opiskeluvaiheessa [A10]. Opiskelijat käyttäisivät tätä toimintatapaa todennäköisesti myös siirryttyään työelämään.

Keskittymällä tehtävien laajuuteen aikamäärien sijaan voidaan vähentää ankkurointivaikutusta työmääräarvioinneissa. Vinouman kannalta on parempi kysyä kuinka paljon vaaditaan asian x tekemiseen (*"How much effort is required to complete X?"*) kuin kysymällä, kuinka paljon saadaan tehtyä y tunnissa (*"How much can be completed in Y work-hours?"*). Jälkimmäisellä tavalla kehittäjät antavat matalampia arvioita [A5].

Arviot ovat myös parempia, kun nykytilannetta verrataan menneeseen, eli kun kysytään kuinka paljon aikaa samankaltaiset tehtävät ovat aiemmin vaatineet [A5]. Arvioita voidaan myös parantaa miettimällä ensin toteutettavan ominaisuuden käyttötapauksen kokoa ja kysymällä sen jälkeen, kuinka monta käyttötapausta projektissa on toteutettu normaalisti sprintin aikana [A5].

Kertominen kognitiivisista vinoumista vähentää vinouman vaikutusta työmääräarvioissa, muttei poista sitä [A8]. Menetelmällä on tehokkaampi vaikutus korkean

ankkurinluvun korjaamiseen kuin matalaan ankkuriin.

Suunnittelupokeri on helppo tapa ehkäistä ankkurointivaikutusta työmääräarvioinnissa [A10, A24]. Tällöin arvioijat eivät voi ankkuroitua toistensa tuottamiin lukuihin, koska menetelmässä arvioijat paljastavat arvionsa yhtä aikaa.

Työmääräarvioinnissa ankkuriluvun tarkkuudella (tarkka tai pyöristetty tuntimäärä) ei ole vaikutusta ankkurointivaikutukseen [A30]. Sen sijaan aikaväleillä arviointi (minimi-maksimi) vahvistaa ankkuroitumista, joten sitä tulisi välttää työmääräarvioinneissa. Ankkurin lähteen luotettavuudella ei ole vaikutusta, eli jopa epäluotettavien lähteiden esittämät epäuskottavat luvut voivat toimia ankkurina. Myös mikä tahansa esillä oleva luku voi toimia ankkurina. Varmin tapa ehkäistä ankkurointivaikutusta on varmistaa, ettei mikään luku ole läsnä työmäärien arviointitilanteessa, kuten esimerkiksi asiakkaan odotukset kustannuksista [A30].

Pariohjelmoinnissa tapahtuva vuorovaikutus voi vähentää ankkurointivaikutusta silloin, kun osapuolilla on eri lähestymistapoja ongelman ratkaisemiseen ja kun he huomaavat puutteita toistensa ratkaisuissa [A7]. Aivoriihitekniikalla suoritetussa ongelmanratkaisussa osapuolet eivät välttämättä tyydy ensimmäiseen mieleen tulevaan ratkaisuun, vaan hiovat ratkaisua yhdessä.

4.6.2 Optimismiharha

Suunnittelupokeri voi vähentää liian optimistisia ja tuottaa tarkempia työmääräarvioita yksilöiden tuottamien arvioiden mekaanisesti laskettuun keskiarvoon verrattuna [A24].

Jos ohjelmistosuunnittelijoita pyydetään ensin arvioimaan työmäärä ihanteellisissa olosuhteissa ja sen jälkeen häntä muistutetaan ihanteellisten ja realististen olosuhteiden erosta ja pyydetään tekemään arvio uudelleen (realistisissa olosuhteissa), tuloksena on vähemmän optimistisia arvioita kuin mitä perinteinen arviointiprosessi tuottaa [A25].

Ryhmissä keskustellen tehty työmääräarviointi tuottaa vähemmän optimistisia arvioita kuin yksin tehdyt [A26]. Ne ovat myös lähempänä toteutuneita työmääriä kuin yksittäisten arvioijien mekaanisesti yhdistetyt arviot. Yksi mahdollinen syy tähän on se, että ryhmät pystyvät tunnistamaan useampia projektiaktiviteetteja ja keskustelemaan oletuksista ja perusteluista arvioiden takana.

Jos projektin riskienhallinnassa riskit esitetään abstrakteina, se lisää optimistisuutta verrattuna siihen, jos riskit on esitetty konkreettisina kuvauksina [A23]. Täten projektin riskejä tulisi arvioida konkreettisten ja yksityiskohtaisten kuvausten avulla joiden myötä mieleen on helpompi palauttaa tietynlaisia projektin onnistumista haittaavia ongelmia.

4.6.3 Itsevarmuusharha

Ohjelmistotuotantoprojektien kustannusarvioinnissa tavataan olla liian itsevarmoja, eli arvioiden uskotaan olevan tarkempia mitä ne todellisuudessa ovat [A27]. Todenmukainen kustannusarvio parantaa projektin budjetointia ja suunnittelua. Intuitiivisesti tehdyistä työmääräarvioista ollaan keskimäärin liian itsevarmoja, joten arvioinnissa tulisi käyttää apuna strukturoituja prosesseja ja työkaluja.

Pariohjelmoinnissa ja aivoriihessä kehittäjät voivat havaita puutteita toistensa ajattelussa ja tiedoissa, mikä voi vähentää liiallista itsevarmuutta [A7].

4.6.4 Kehysvaikutus

Kun ominaisuudet esitettiin vaatimuksina, koehenkilöt tuottivat vähemmän luovia mutta käytännöllisempiä ratkaisuja ja suunnitelmia verrattuna siihen, jos ne esitettiin ideoina [A19]. Sana ”vaatimus” rajoittaa osallistujan luovuutta ja saa päätöksentekijän uskomaan, että ongelma on hyvin ymmärretty ja pääosin ratkaistu. Kun ominaisuuden kehittämisessä toivotaan luovuutta, niiden esittämistä vaadittuina tulisi välttää ja sen sijaan korostaa niiden tärkeysastetta, prioriteettia ja varmuutta [A19]. Vastaavasti jos ominaisuuden suunnittelussa halutaan käytännöllisempiä ratkaisuja, ominaisuuksiin tulisi viitata vaatimuksina [A20]. Termi ”vaatimus” tulisi siis varata vain sellaisille ominaisuuksille, joilla on korkea tärkeysaste ja korkea varmuus [A19].

4.6.5 Vahvistusharha

Ohjelmissa esiintyviä virheitä löytyy helpommin, kun testaaja pyrkii aktiivisesti kyseenalaistamaan ominaisuuden toimivuuden, ts. kysyy itseltään mitä virheitä toteutuksessa saattaisi olla. Koodivirheiden etsinnässä häiriön syy löytyy nopeammin, kun pyrkii perustelemaan miksi epäilty syy ei ole ongelman lähde [A12].

Tutkimusten mukaan loogisen päättelykyvyn ja kriittisen ajattelun jatkuva käyttö ja harjoittaminen vähentävät vahvistusvinoumaa. Tietojenkäsittelyn opiskelijat menestyivät paremmin heille annetussa päättelytehtävässä, ns. Wasonin korttitehtävässä, kuin työelämässä ohjelmistofirmoissa työskentelevät [A15]. Loogista päättelyä ja matemaattisia todistustekniikoita opiskelleilla on havaittu esiintyvän vähemmän vahvistusvinoumaa, eli he pyrkivät todennäköisemmin osoittamaan väitteen vääräksi kuin hyväksymään sen oikeaksi [A14]. Kokeneet mutta epäaktiiviset ohjelmistokehittäjät ja testaajat (nykyisin tutkimustyössä olevat) menestyivät paremmin vahvistusvinoumaa mittaavissa testeissä, kuin kokeneet aktiiviset kehittäjät ja testaajat [A14].

Myös tutkijoilla esiintyy vähemmän vahvistusvinoumaa kuin ohjelmistokehityksessä ja testauksessa aktiivisesti työskentelevillä [A16]. Tutkimuksen kirjoittajat arvelivat tämän johtuvan analyyttista ja kriittistä ajattelua vahvistavista teoreettisista tietojenkäsittelytieteen kursseista sekä aktiivisesta osallistumisesta tutkimustyöhön,

joka vaatii hypoteesin testaustaitoja. Säännöllinen abstrakti päättely ja hypoteesien testaus siis vaikuttaa vahvistusvinouman tasoon [A16].

Vahvistusvinoumasta johtuvaa positiivisiin testitapauksiin keskittymistä voidaan vähentää tekemällä määrittelyistä tarkempia [A17]. Vinoumaa esiintyy testaajilla vähemmän, kun määrittelyihin lisätään negatiivisia ekvivalenssiluokkia, sillä silloin testaajien on helpompi päätellä negatiiviset testitapaukset. Myös kokemus vähentää testauksessa esiintyvää vahvistusvinoumaa, joten testaajia tulisi kouluttaa vinouman olemassaolosta ja negatiivisten testien tärkeydestä [A18]. Kolmas keino vinouman vähentämiseen on kannustaa kehittäjiä tekemään negatiivinen testi jokaista positiivista testiä kohden [A17].

4.6.6 Edustavuusharha

Vanhentuneiden vaatimusten olemassaolo määrittelydokumenteissa voi vaikuttaa työmääräarvioihin. Kun osallistujia pyydettiin jättämään ylimääräiset vaatimukset huomiotta, he antoivat tehtäville suurempia työmääräarvioita [A28]. Eli kaikenlainen ylimääräinen tehtävään liittyvä informaatio voi johtaa korkeampiin työmääräarvioihin.

4.6.7 Saatavuusharha

Aiempi tietämys dokumentaation sisällöstä tehostaa tiedonhakua, mutta se voi myös johtaa vajavaiseen hakuprosessiin, koska yksilöillä on tiettyjä odotuksia siitä, miten ja mistä he löytävät etsimänsä tiedon. Odotukset voivat olla vääriä, joka voi johtaa tehottomaan dokumentaation käyttöön. Hakuprosessia voidaan parantaa avainsanoilla ja rakenteella [A13]. Ontologiaan pohjautuva dokumentaatio mahdollistaa informaation välisten yhteyksien ja riippuvuuksien esittämisen ja siten tehostaa dokumentaation käyttöä.

4.6.8 Not-invented-here -ilmiö

Not-invented-here eli NIH-ilmiö vähentää kehittäjien halua käyttää ulkoisia teknologioita, ideoita ja tietoa, ja näkyy yrityksen sisäisen kehitystyön suosimisena [A22]. Vinouman vaikutuksen vähentämisen keinoiksi mainitaan yrityksen kulttuurin kehittäminen sekä prosessien ja suorituskyvyn mittaamisen kehittäminen, kun pohditaan artefaktien ulkoisen ja sisäisen kehityksen välillä, ja ulkoisten artefaktien käytöstä palkitseminen silloin kun niistä on hyötyä yrityksen kilpailukyvyn kannalta.

4.6.9 Valintaharha

Valintaharha on heikompi, jos työmäärä- ja kustannusarviot ovat tarkempia. Niitä voidaan parantaa paremmilla arviointiprosesseilla, historiatietojen paremmalla

käytöllä, kehitettävän ohjelmiston paremmalla ymmärtämisellä ja paremmalla kommunikaatiolla asiakkaan kanssa [A29].

5 Analyysi

Luvussa analysoidaan tutkimuksen tuloksia ja vastataan tutkimuskysymyksiin. Ensin analysoidaan sitä, miten hyvin aineistossa hyödynnettiin alkuperäisiä psykologian tutkimuksia. Sitten pohdintaan eri vinoumien vaikutuksia ohjelmistotuotannossa ja analysoidaan aineistosta löytyneitä vähentämiskeinoja. Lopuksi tehdään yhteenveto löytyneistä vinoumien vaikutuksista ja niihin sopivista vähentämiskeinoista.

5.1 Psykologian tutkimuksen hyödyntäminen

Suurimmassa osassa julkaisuista oli viittaus alkuperäiseen psykologian lähteeseen vinouman määrittelyssä, joten viittaukset olivat aineistossa hyvällä tasolla.

Sen sijaan vinoumien taustamekanismeja selitettiin huonosti. Ainoastaan alle puolessa julkaisuista pyrittiin selittämään taustamekanismia, ja selitykset olivat suurimmaksi osaksi melko pinnallisia. Esimerkiksi heuristiikkojen mainitseminen taustamekanismina ei juuri selitä vinouman taustalla esiintyviä ajatusprosesseja. Myös ongelmanratkaisun yksinkertaistaminen ja kognitiivisen kuorman minimointi ovat hyvin yksinkertaistettuja ja abstrahoituja kuvauksia aivojen toiminnasta. Selitykset koskivat yleisemmin kaikkien kognitiivisten vinoumien taustamekanismia. Ainoastaan edustavuus- ja saatavuusharhan sekä ankkurointivaikutuksen taustamekanismeja selitettiin yksityiskohtaisemmin.

Psykologian teorioita hyödynnettiin aineistossa yleensä ottaen hyvin. Ainostaan kolme julkaisua ei viitannut psykologian teorioihin lainkaan. Monissa julkaisuissa teoriat otettiin huomioon siten, että niissä selitettiin vinouman tutkimukseen liittyviä tutkimustuloksia. Muutamassa julkaisussa suoritettu koe perustui psykologian tutkimuksessa tehtyihin kokeisiin.

5.2 Vinoumien vaikutus ohjelmistotuotannossa

Seuraavaksi analysoidaan kognitiivisten vinoumien vaikutuksia ohjelmistotuotannossa ja pohditaan, mitä seurauksia niillä voi olla ohjelmistokehityksessä. Vinoumista raportoitiin enimmäkseen haitallisia vaikutuksia.

5.2.1 Vinoutuneet työmääräarviot

Kun työmääräarviot eivät vastaa toteutunutta ajankäyttöä, se voi johtua liiasta itsevarmuudesta omiin kykyihin, ylimääräisestä tiedosta dokumentaatiossa sekä kaikista esillä olevista luvuista, jotka voivat vaikuttaa arvioihin. Työmääräarvioinnissa esiintyviä kognitiivisia vinoumia ovat ankkurointivaikutus, optimismiharha ja itsevarmuusharha.

Kaikki ylimääräinen tieto dokumentaatiossa voi aiheuttaa suurempia työmääräarvioita. Mitä laajempi dokumentaatio on, sitä suuremmalta ja monimutkaisemmalta järjestelmä vaikuttaa. Työmääriä arvioitaessa päätöksentekijät voivat ankkuroitua mihin tahansa näkyvillä olevaan lukuun. Ilmeisesti kaikenlainen ylimääräinen informaatio voi vääristää käsitystä ohjelmiston koosta ja monimutkaisuudesta. Tutkimus ei kuitenkaan kerro, voiko liian suppea dokumentaatio vastaavasti johtaa matalampiin työmääräarvioihin, koska järjestelmä voitaisiin tulkita yksinkertaisempana kuin todellisuudessa.

Ankkurointivaikutus on voimakkaampi, kun työmääriä pilkotaan, kuin jos käytetään koodiriveihin tai toimintopisteisiin perustuvia menetelmiä. Syynä on luultavasti tehtävien pilkkominen pienempiin osiin, jolloin pienet tehtävät näyttäytyvät nopeina ja helppoina tehdä eikä osata nähdä, että kokonaisuus onkin suurempi ja monimutkaisempi kuin pienet, helpot osaset.

Kun tarkastellaan koodirivejä tai toimintopisteitä, kehittäjät luultavasti alkavat miettiä mitä kaikkea vaaditaan, että ominaisuus saadaan koodattua, ja miettivät perusteellisemmin mitä ominaisuuden toteuttaminen vaatisi. Tällöin ominaisuuden kokoa aletaan miettiä konkreettisesti, jolloin liiallinen optimismi arvioinnissa vähenee.

Myös kysymyksen muotoilulla on väliä työmääräarvioinnissa. Kun arvioijat pohtivat, kuinka paljon voidaan toteuttaa tietyssä ajassa, työmääräarviot ovat optimistisempia, kuin jos keskitytään siihen, mitä tehtävän toteuttaminen vaatii. Ajanjaksoon fokusoituminen aiheuttaa siis sen, että arvioijat eivät pohdi kaikkia tehtävän toteutukseen vaadittavia asioita niin hyvin kuin jos keskitytään tehtävän laajuuteen.

Edellä mainitut seikat siis viittaisivat siihen, että kognitiivisten vinoumien vaikutus voimistuu silloin, kun työmääräarvioinnissa fokus on muualla kuin tehtävien laajuudessa ja kaikissa niissä seikoissa, joita tehtävän toteutukseen vaaditaan. Keskittymällä ominaisuuden toteuttamiseen vaadittavaan työn ja koodin määrään saadaan realistisempia työmääräarvioita.

Yksityiskohtiin keskittymisen vaikutus optimismiharhaan näkyy myös riskienhallinnassa: mitä konkreettisemmalla tasolla riskejä tarkastellaan, eli mitä yksityiskohtaisemmin riskejä kuvataan, sitä realistisempi käsitys niiden hallinnasta on. Abstrakteja riskejä tarkastellessa ei todennäköisesti osata ottaa kaikkia siihen liittyviä tekijöitä huomioon, joten luottamus niiden hallintaan on optimistisempi.

5.2.2 Omien tuotosten yliarviointi ja teknologiavalintojen vinoutuminen

Projektissa käytettävien teknologioiden, kehitystyökalujen ja koodikirjastojen valinnassa voi esiintyä valintaharhaa ja NIH-ilmiötä, jotka vääristävät optimaalisimman vaihtoehdon päättämistä. Valintaharhan takia päätöksentekijät keskittyvät liikaa johonkin mieluisaan ominaisuuteen, jolla ei loppujen lopuksi ole välttämättä suurta merkitystä optimaalisen valinnan kannalta.

NIH-ilmiö aikaansaa sen, että kehittäjät pitävät tiiminsä kykyjä ylivertaisina muihin

nähden, mikä näkyy vastahakoisuutena hyödyntää ulkoisia ideoita ja ohjelmistokomponentteja. Sisäisesti kehitettyjä ideoiden ja komponenttien arvoa yliarvioidaan ja muiden tuottamien ideoiden hyötyjä aliarvioidaan.

Ikea-efekti saa ohjelmistokehittäjät kiintymään itse tehtyihin ohjelmistoartefakteihin sitä enemmän, kun niiden määrittelyyn ja toteutukseen käytetään aikaa ja vaivaa. Tällöin niiden tärkeys ja merkittävyys kasvaa kehittäjiensä silmissä. Ikea-efekti saattaa olla kytköksissä NIH-ilmiöön. Kun kehittäjät käyttävät aikaa ideoiden tai komponenttien kehittämiseen, he kiintyvät niihin ja alkavat pitää niitä arvokkaampina kuin muualla kehitettyjä komponentteja. Kun sisäisesti kehitettyjä komponentteja pidetään arvokkaampina kuin ulkoisesti kehitettyjä, ulkoisia komponentteja ei haluta käyttää.

Kehitystyökaluja ja komponentteja valitessa tietyillä vaihtoehdoilla nähdään positiivisia ominaisuuksia, jotka saavat nämä vaihtoehdot näyttämään kehittäjän silmissä entistä mieluisammilta. Vastaavasti niiden vaihtoehtojen, joilta nämä tietyt mieluisat ominaisuudet puuttuvat, negatiiviset puolet korostuvat mikä johtaa siihen, että näitä vaihtoehtoja ei haluta tosissaan edes harkita. Asenne voi olla olemassa jo ennen päätöksentekotilannetta, jolloin kaikki vaihtoehdot eivät saa samanarvoista mahdollisuutta ja päätöksenteko vinoutuu. Sen seurauksena kehittäjät saattavat päätyä valitsemaan vaihtoehtoja, jotka eivät ole ohjelmiston tai projektin kannalta parhaita mahdollisia ratkaisuja.

5.2.3 Kokemuksesta hyötyä ja haittaa

Kokeneet ohjelmistokehittäjät tuottavat kokemattomia todennäköisemmin useita ratkaisuja käsillä olevaan ongelmaan ja pyrkivät parantelemaan niitä löytääkseen parhaan mahdollisimman ratkaisun. Kokemattomat kehittäjät ankkuroituvat tavallisesti ensimmäiseen mieleen tulevaan ratkaisuun. Tämä johtuu todennäköisesti siitä, että kokeneilla kehittäjillä on enemmän kokemusta erilaisista toteutustavoista ja niiden sopivuudesta eri tilanteisiin, jolloin heidän arviointikykynsä ansiosta he pystyvät tekemään parempia päätöksiä.

Kokemuksesta voi olla myös joissakin tilanteissa haittaa. Kokeneet ohjelmistokehittäjät eivät ole yhtä halukkaita tutkimaan dokumentaatiota kuin kokemattomat kehittäjät. He ovat itsevarmempia siitä, että he tietävät tarvittavat asiat eivätkä siksi tarkista tietoja dokumentaatiosta. Tällöin tärkeääkin informaatiota voi jäädä havaitsematta ja ottamatta huomioon, joka voi johtaa väärinymmärryksiin ohjelmiston vaatimuksista.

5.2.4 Dokumentaation ja koodin alueita voi jäädä huomiotta

Saatavuus- ja edustavuusharha saavat kehittäjät kiinnittämään enemmän huomiota usein käytettyihin ja mieleenpainuvampiin koodin tai dokumentaation osiin. Tällöin muut osat jäävät vähemmälle tarkastelulle ja siten mahdollisesti vähemmälle ylläpidolle, jolloin koodi tai dokumentaatio saattaa vanhentua. Sen seurauksena ohjel-

miston tekninen velka voi kasvaa. Vanhentuneesta dokumentaatiosta ei ole hyötyä, ja jos sen perusteella kehitetään ohjelmistoa, seurauksena voi syntyä puutteellisten määrittelyn pohjalta toteutettuja ominaisuuksia, joiden korjaamiseen kuluu taas resursseja.

Vinoumien aiheuttamia dokumentointiin liittyviä ongelmia voivat olla oletusten ja aiemman tietämyksen aikaansaama dokumentaation tehoton käyttö. Vaikka aiempi tietämys tehostaa tiedonhakua, koska henkilö uskoo tietävänsä mistä vastaus löytyy, se voi johtaa dokumentaation tarkastelijaa harhaan, jos olettamukset ovat virheellisiä tai vanhentuneita. Tiettyjä dokumentaation osia saattaa jäädä huomiotta, koska haetun tiedon oletetaan löytyvän muualta. Tällöin tarvittavaa informaatiota ei joko löydy tai sen etsimiseen kuluu enemmän aikaa, mikä aiheuttaa turhautumista ja ennestään vähentää motivaatiota pitää dokumentaatio ajan tasalla. Pahimmassa tapauksessa turhautuminen johtaa noidankehään, jossa dokumentaatiota ylläpidetään ja käytetään yhä vähemmän, mikä voi luoda kielteistä suhtautumista dokumentointiin ylipäätään.

5.2.5 Puutteellinen testaus

Vahvistusharha ilmenee testauksessa positiivisten testien kirjoittamisen suosimisena ja negatiivisten testitapausten vähäisempänä määränä. Testaajat siis pyrkivät vahvistamaan, että ohjelma toimii vaatimusten mukaisesti. Negatiiviset testitapaukset ovat tärkeitä testikattavuuden kannalta, koska ne voivat paljastaa ohjelmasta vikoja, joita positiivisilla testeillä ei välttämättä pystytä löytämään.

Kun kehittäjien vahvistusharhan taso on matala, he kirjoittavat enemmän negatiivisia testejä ja tällöin ohjelmassa on vähemmän vikoja. Korkea vahvistusharhan taso on siis yhteydessä suurempaan ohjelmointivirheiden määrään. Kun ohjelmistossa on paljon vikoja, ohjelmiston laatu on heikompi ja käyttäjät ovat vähemmän tyytyväisiä, koska silloin ohjelmisto on altis toimintahäiriöille. Tällöin ohjelmiston virheiden etsintään ja korjaamiseen kuluu aikaa ja kustannukset lisääntyvät.

5.2.6 Koodin huolimaton uudelleenkäyttö

Kun koodia tai SQL-kyselyjä käytetään uudelleen, ylimääräiset ominaisuudet tai virheet voivat siirtyä uuteen kohteeseen. Kun olemassa oleva koodi toimii alkuperäisessä kontekstissa, sen oikeellisuudesta ja toimivuudesta myös uudessa kontekstissa ollaan itsevarmoja.

Koodin, SQL-kyselyjen tai komponenttien uudelleenkäytön etuna on ajan säästyminen, koska koodia ei tarvitse tehdä alusta alkaen kokonaan itse. Siksi niitä on houkuttelevaa ja usein myös projektin etenemisen kannalta tehokasta käyttää. Mutta jos uudelleenkäytettävästä koodista paljastuukin vikoja, niiden korjaamiseen kuluu aikaa. Jos viat ovat vaikeasti havaittavia tai johtavat harvoin ohjelmistohäiriöön, niiden löytämiseen saattaa kulua paljon aikaa ja ne saattavat ehtiä aiheuttaa haittaa ohjelmiston käyttäjälle tai asiakkaalle.

5.3 Vinoumien vähentäminen

Seuraavaksi analysoidaan aineistosta löytyneiden kognitiivisten vinoumien vähentämiskeinoja. Vähentämiskeinoja löytyi lähes kaikkiin vinoumiin. Monet keinot olivat kuitenkin ainoastaan ehdotuksia, eli tieteellistä näyttöä niiden pätevyydestä tai tehokkuudesta ei ollut.

5.3.1 Ankkurointivaikutuksen vähentäminen työmääräarvioinnissa

Työmääräarvioinnissa kognitiiviset vinoumat saavat arvioijat keskittymään arvioinnin kannalta vääriin asioihin tai ankkuroitumaan muiden tuottamiin tai muihin esillä oleviin lukuihin, jolloin tehdyt työmääräarviot ovat matalampia tai korkeampia todellisiin työmääriin verrattuna.

Työmääräarvioinnissa ankkurointivaikutusta voidaan vähentää suunnittelupokerilla. Koska arvioijat eivät tiedä muiden työmääräarvioita ennen kuin he tekevät arvion itse, he eivät voi ankkuroitua muiden lukuihin, vaan tekevät arvion omien päätelmiensä pohjalta. Silti ympäristössä mikä tahansa projektiin liittymätönkin esillä oleva luku voi toimia ankkurina, joten työmääräarviointeja tehdessä tulee pitää huolehtia siitä ettei ympäristössä ole esillä minkäänlaisia lukuja.

Pyydettäessä työmääräarviota luvun tarkkuudella ei ole väliä ankkurointivaikutuksen määrään. Sillä ei ole siis väliä, pyydetäänkö tarkkaa tai pyöristettyä tuntimäärää.

Työmääräarvioinnissa kannattaa keskittyä tehtävän laajuuteen, eikä siihen kuluvaan aikaan, kuten esimerkiksi että ehditäänkö toteuttaa jokin ominaisuus tietyssä ajassa. Kuluvaan aikaan keskittyminen vie huomiota tehtävän konkreettisista yksityiskohdista. Kun arvioidaan ominaisuuden toteuttamiseen vaadittavaa työmäärää ja keskitytään sen yksityiskohtiin, koodiriveihin tai toimintopisteisiin, tuloksena on realistisempia ja vähemmän vinoutuneita työmääräarvioita.

Ohjelmistokehittäjiä olisi hyvä myös valistaa ankkurointivaikutuksesta ja niiden vaikutuksesta. Vaikka valistamisella ei ole tyypillisesti saatukaan merkittäviä hyötyjä, olisi luultavasti silti hyödyllistä muistuttaa päätöksentekijöitä ankkurointivaikutuksesta joka kerta kun aletaan tehdä arvioita.

5.3.2 Optimismiharhan vähentäminen työmääräarvioinnissa

Työmääräarvioinnissa liiallista optimismia voidaan vähentää tekemällä arviot ryhmissä. Keskustellen tehdyt arviot ovat lähempänä totuutta kuin yksittäiset arviot, koska keskustelun kautta arvioijat kykenevät tunnistamaan useampia työn toteutukseen vaadittavia tekijöitä ja aktiviteetteja. Työmääräarviot tarkentuvat, kun arvioijat voivat keskustella oletuksista ja perusteluista arvioiden takana.

Toinen tapa vähentää optimismiharhaa ja tarkentaa työmääräarvioita on pyytää kaksi työmääräarviota: arvio ihanteellisissa olosuhteissa, jossa kaikki sujuu täydellisesti, ja toinen arvio huonoissa olosuhteissa, jossa kohdataan paljon vastoinkäymisiä.

Arvioijia pyydetään siis miettimään, miten asiat etenisivät kummassakin skenaariossa. Esimerkiksi huonoissa olosuhteissa arvioijien tulisi kysyä tai keskustella siitä, mitä kaikkea voisi tapahtua ja mitä kompastuskiviä ei ehkä osata ottaa huomioon. Realistinen työmääräarvio on todennäköisesti jossain ihanteellisen ja pessimistisen arvion välillä.

5.3.3 Edustavuusharhan vähentäminen

Edustavuusharhaa voidaan vähentää pitämällä määrittelydokumentit ajan tasalla, koska kaikenlainen ylimääräinen informaatio voi johtaa korkeampiin työmääräarvioihin. Voisi olla mielenkiintoista selvittää, voisiko vastaavasti puutteellinen tai tiivis dokumentaatio johtaa matalampiin työmääräarvioihin, koska ohjelmisto saatettaisiin nähdä pienempänä ja yksinkertaisempana kuin todellisuudessa.

5.3.4 Vahvistusharhan ja itsevarmuusharhan vähentäminen

Omien päätelmien kyseenalaistaminen voi vähentää vahvistusharhaa ohjelmointivirheiden etsinnässä ja optimaalisen ratkaisun etsimisessä. Kun virheen syytä etsiessä lähtee ensin rajaamaan pois vikaan liittymättömiä tekijöitä pois pitäen samalla useita mahdollisuuksia avoinna, on vähemmän todennäköistä, että testaaja keskittyy ainoastaan yhteen johtolankaan ja etsii todisteita ainoastaan vahvistaakseen uskomuksensa.

Etsiessään ratkaisua johonkin ongelmaan, kehittäjien tulisi aktiivisesti kyseenalaistaa ratkaisujensa toimivuutta ja sopivuutta kyseiseen tilanteeseen. Vahvistusharha sekä itsevarmuusharha heikkenevät, kun ei perustele itselleen miksi ratkaisu toimii, vaan pyrkii miettimään, mitä heikkouksia ratkaisussa voi olla.

Testauksessa vahvistusharhan heikentämistä ja parempaa testikattavuutta voidaan tavoitella tekemällä kattavampia määrittelyjä, jotka sisältävät myös negatiiviset ekvivalenssiluokat, kouluttamalla vinoumista ja negatiivisten testien tärkeydestä, sekä kannustamalla tekemään yksi negatiivinen testitapaus jokaista positiivista testitapausta kohden.

5.3.5 Vinoumien vähentämiskeinojen luokittelu

Keinot vinoumien vähentämiseen voidaan kategorisoida sen mukaan ketä tai mitä kohtaan vähentämismenetelmää käytetään. Vähentämiskeinoja voidaan kohdistaa päätöksentekijöihin tai tehtäviin. Apuna voidaan käyttää myös teknisiä menetelmiä tai voidaan pyrkiä muokkaamaan ympäristöä siten, että ihmisen luontaiset ajatusprosessit sopivat käsillä olevan ongelman ratkaisemiseen paremmin. Tällöin päätöksentekijöiden käyttämät kognitiiviset oikotiet tuottavat intuitiivisesti optimaalisempia ratkaisuja vinoumien sijaan.

Jos vähentämiskeinot kohdistetaan **päätöksentekijään**, tällä on vastuu havaita tilanteet, joissa vinoumia voi tapahtua ja tehdä sitten sopivia muutoksia omaan käyt-

täytymiseensä. Tilanteiden tunnistamista, joissa vinoumia tavallisesti esiintyy, voidaan opettaa päätöksentekijöille esimerkiksi kursseilla tai koulutuksella.

Ongelmanratkaisutilanteessa, jossa on mahdollista tuottaa useita ratkaisuja, päätöksentekijöiden ei tulisi tyytyä ensimmäiseen mieleen tulevaan ratkaisuun, vaan jatkaa ongelma-alueen tutkimista ja pyrkiä tuottamaan useampia ratkaisuja. Tällöin todennäköisyys optimaalisimman ratkaisun löytymiseen nousee. Omiin päätelmiin tulisi suhtautua kriittisesti ja päätöksentekijöiden pitäisi pystyä perustelemaan mitä heikkouksia kussakin ratkaisussa voisi olla. Tällöin itsevarmuus niiden oikeellisuudesta vähenee ja niihin pystytään suhtautumaan objektiivisemmin.

Tekemällä päätöksiä pareittain tai ryhmissä voidaan myös vähentää vinoumia. Tällöin ryhmän jäsenet voivat havaita epäloogisuuksia ja puutteita toistensa päätelmissä. Pariohjelmointi ja aivoriihitekniikka ovat eräitä keinoja yhdessä tapahtuvaan asioiden tarkasteluun monesta näkökulmasta toisten puutteita paikaten.

Tehtäviä voi muokata pitämällä dokumentaation ajan tasalla ja huolehtia siitä, että määrittelyt ovat mahdollisimman tarkkoja. Ajantasainen dokumentaatio ja tarkat määrittelyt parantavat työmääräarviointia, koska ne mahdollistavat päätöksentekijöiden tehdä tarkempia arvioita. Tarkempien määrittelyjen avulla testaajat voivat myös testata ohjelmistoja kattavammin. Työmääräarviointia voidaan muokata tekemällä kaksi arviota ja valitsemalla arvion niiden välistä.

Ajatteluprosesseja voidaan muokata kiinnittämällä huomiota sanamuotojen valintaan ja siihen, miten tehtävät kehystetään. Vaatimusmäärittelyssä sanan *vaatimus* käyttö rajoittaa luovuutta, kun taas sanan *idea* käyttäminen saa kehittäjät näkemään enemmän vaihtoehtoja. Päätöksentekijät ovat myös vähemmän optimistisia kyvystään hallita riskejä, kun ne esitetään konkreettisina eikä abstrakteina. Työmääräarvioinnissa voidaan keskittyä tehtävän laajuuteen, eikä niinkään siihen mitä ehditään tehdä tietyssä ajassa. Tällöin päätöksentekijä ottaa huomioon tehtävän yksityiskohdat, mikä tekee työmääräarvioista realistisempia.

Ehdotuksia vinoumien vähentämiseen, jotka voidaan lajitella **teknisiksi menetelmiksi**, olivat ohjelmiston jäljitettävyystietojen käyttö, suunnittelupokeri ja ontologiaan perustuva dokumentaatio.

Ohjelmiston jäljitettävyystietojen käytöllä ohjelmistokehittäjät voivat muodostaa kattavamman kuvan järjestelmän tilasta, jolloin ankkuroituminen ensimmäiseen ymmärrykseen järjestelmästä vähenee. Suunnittelupokeri estää ankkuroitumisen muiden tuottamiin arvioihin. Ontologiaan perustuva dokumentaatio vähentää saatavuusharhaa luomalla yhteyksiä käsitteiden välille. Tällöin dokumentaation rakenne ohjaa dokumentaation lukijaa löytämään tietoa, eikä hänen siten tarvitse luottaa omiin oletuksiin ja käsityksiinsä tiedon sijainnista, jotka voivat olla virheellisiä tai vanhentuneita.

5.4 Yhteenveto vinoumista ja niiden vähentämiskeinoista

Kaikkiin vinoumiin ei löytynyt vähentämiskeinoja. Keinot on kuvattu taulukossa 14. Joitakin vähentämiskeinoja voisi soveltaa myös muihin tilanteisiin kuin aineistossa esitettyihin vinoumiin: esimerkiksi kokemattomien ohjelmistokehittäjien vinoumia voitaisiin hillitä pariohjelmoinnilla, koska tällöin kokeneempi voi olla kokemattoman tukena eri vaihtoehtojen tuottamisessa ja analysoimisessa. Omien päätelmien kyseenalaistaminen ja niiden heikkouksien pohtiminen voisi myös toimia monessa eri päätöksentekotilanteessa, vaikka se vaatiikin ymmärrystä pysähtyä ja vaivannäköä omien ratkaisujen parantelemiseksi. Kirjoittajan ehdottamat vinoumien vähentämiskeinot ovat taulukossa kursiivilla.

Ikea-efektiin ei löytynyt aineistosta vähentämismenetelmiä. Vinoumaa voitaisiin mahdollisesti vähentää määrittämällä ominaisuuden tärkeys yhdessä asiakkaan kanssa jo määrittelyvaiheessa. Vaikka ominaisuuden toteutukseen käytetty aika lisäisikin sen tärkeyttä kehittäjiensä silmissä, niin alussa määritelty tärkeysaste muistuttaisi heitä ominaisuuden todellisesta tärkeysasteesta.

Kuten aiemmin pääteltiin, työmääräarvioiden optimistisuuden vähentämiseen sopivia menetelmiä havaittiin olevan keinot, jotka saavat arvioijat pohtimaan kaikkia niitä konkreettisia yksityiskohtia, mitä tehtävien toteutukseen vaaditaan. Tehtävien laajuuden arviointi mahdollisimman matalalla tasolla, jopa koodirivi- tai toimintopistetasolla, voisi toimia vinoumaa vähentävänä keinona.

Saatavuus- ja edustavuusharha voivat aiheuttaa sitä, että usein tarkasteltua ja helposti muistettavaa koodia tarkastellaan useammin. Ohjelmistoartefaktien elinkaaren jäljitettävyystietoja voitaisiin hyödyntää järjestelmän historiatietojen esittämisessä: tietojen avulla voitaisiin nähdä mihin koodin osiin on tehty muutoksia eniten ja mitkä ovat jääneet vähemmälle huomiolle ja mahdollisesti tekniselle velalle.

Taulukon viimeistä vinoumaa, tarjouskilpailuiden valintaprosessin vinoutumista, on hankala estää järjestelmän toimittajan näkökulmasta, sillä valintaharha esiintyy tässä tapauksessa asiakkaalla. Vinouman vähentäminen jää siis asiakkaan harteille – keinona on vähentää hinnan merkitystä ja painottaa vaihtoehtojen muita ominaisuuksia.

Vinouma	**Vaikutus**	**Vähentämiskeino**
Ankkurointivaikutus	Ankkuroituminen mihin tahansa lukuun työmääräarvioinnissa [A2, A30]	Kaikkien lukujen piilottaminen työmääräarvioinnissa [A30] Suunnittelupokeri [A10, A24]
	Toteutettuun ratkaisuun ankkuroituminen koodia tai SQL-kyselyjä uudelleenkäytettäessä; ylimääräisten ominaisuuksien siirtyminen uuteen kohteeseen [A3, A1]	Jäljitettävyystiedot [A9] *Toteutuksen tarkistaminen ja kyseenalaistaminen*
Vahvistusharha	Vähäinen kokemus ohjelmistosuunnittelusta [A7]	*Pariohjelmointi*

	Kokemus ohjelmistosuunnittelusta vähentää halua tutkia dokumentaatiota [A7]	*Omien oletusten kyseenalaistaminen*
Edustavuusharha	Määrittelydokumentaatiossa olevan ylimääräisen informaation vaikutus työmääräarvioihin [A28]	Ylimääräisen informaation välttäminen dokumentaatiossa [A28] *Dokumentaation pitäminen ajan tasalla*
Ikea-efekti	Ominaisuuksien pitäminen alkuperäistä tärkeämpänä kun sen määrittelyyn ja toteutukseen käytetään aikaa; haluttomuus tarvittaessa luopua niistä esim. valitessa ominaisuuksia tuotantoon [A21]	*Ominaisuuden tärkeyden arviointi määrittelyvaiheessa asiakkaan kanssa*
Itsevarmuusharha	Liiallinen itsevarmuus työmääräarvioiden paikkansapitävyydestä [A2]	Prosessien ja työkalujen käyttö arvioinnin apuna [A27]
NIH-ilmiö	Yrityksen ulkopuolelta tulevien ideoiden ja teknologioiden hyötyjen aliarviointi ja haluttomuus hyödyntää niitä [A22]	Yrityksen kulttuurin kehittäminen; prosessien ja suorituskyvyn mittaaminen ohjelmistoartefakteja valitessa; ulkoisten teknologioiden käytöstä palkitseminen [A22]
Optimismiharha	Optimistiset työmääräarviot [A2]	Työmääristä kaksi arviota, ihanteellisissa ja realistisissa olosuhteissa [A25] Työmääräarvioista keskusteleminen [A26] Aiemmin toteutettujen vastaavanlaisten ominaisuuksien toteutuneiden työmäärien hyödyntäminen [A5] *Keskitytään tehtävän toteuttamisen vaatimiin yksityiskohtiin*
	Optimistinen hallinnan tunne riskeistä kun projektin riskit ymmärretään hyvin tai kun riskit on kuvattu korkealla tasolla [A23]	Riskien pohtiminen konkreettisella ja yksityiskohtaisella tasolla [A23]

Kehysvaikutus	Määriteltyjen ominaisuuksien lopullisuutena pidetty taso, kun niitä kutsutaan vaatimuksiksi tai ideoiksi [A19]	Ominaisuuksien kehystäminen ideoiksi kun halutaan luovuutta, ja vaatimuksiksi kun ominaisuuksista ollaan varmoja [A19] Määrittämällä ominaisuuden tärkeysaste ja varmuus [A19]
Saatavuusharha	Usein tarkastellun ja helposti muistettavan koodin ja ominaisuuksien tarkempi käsittely [A12]	*Jäljitettävyystiedot tarkastelun apuna kun halutaan tietää mitä koodin osia on käsitelty eniten ja mitkä jäävät vähemmälle huomiolle*
	Aiempi tietämys dokumentaatiosta ohjaa hakua [A13]	Ontologiaan perustuva dokumentaatio [A13]
Vahvistusharha	Positiivisten testitapausten suosiminen [A17, A18]	Tarkemmat määrittelyt testausta varten; negatiivisten ekvivalenssiluokkien lisääminen määrittelyihin [A17] Vahvistusvinoumasta ja negatiivisten testien tärkeydestä muistuttaminen [A18] Negatiivisen testin tekeminen jokaista positiivista testiä kohden [A17]
	Yhden hypoteesin testaaminen kerrallaan ohjelmointivirheiden etsinnässä [A12]	Ohjelmavirheiden etsintä aloitetaan rajaamalla niitä tekijöitä pois, joista tiedetään, etteivät ne aiheuta virhettä [A12]
Valintaharha	Teknologioiden, arkkitehtuurien tai kehitystyökalujen valinta kustannusten tai työmäärän perusteella [A29]	Arviointiprosessien kehittäminen; historiatietojen käyttö; järjestelmän ymmärryksen parantaminen [A29]
	Hinnan korostaminen tarjouskilpailussa vääristää valintaprosessia järjestelmän toimittajasta [A29]	*Vastuu vinouman vähentämisestä asiakkaalla*

Taulukko 14: Vinoumien vaikutuksia ja niiden vähentämiskeinoja.

5.5 Tutkimuksen rajoitukset

Analyysin lopuksi pohditaan tulosten pätevyyttä ja yleistettävyyttä.

Ensinnäkin on mahdollista, että käytetyllä hakustrategialla ei tavoitettu kaikkia olemassa olevia aihetta koskevia artikkeleita. Lisäksi aineiston ulkopuolelle jouduttiin jättämään joitakin potentiaalisia artikkeleita, sillä niihin pääsy oli rajoitettu.

Käytetyt hakulauseet sisälsivät ainoastaan käsitteen *kognitiivinen vinouma*, mutta suurin osa aineiston artikkeleista käsitteli jotain tiettyä kognitiivista vinoumaa. Haun ulkopuolelle on mahdollisesti jäänyt sellaisia artikkeleita, jotka käsittelevät jotain tiettyä vinoumaa, mutta joissa ei käytetä kognitiivisen vinouman termiä. Hakulauseet ja käytetyt tietokannat kattoivat kuitenkin todennäköisesti tärkeimmät julkaisut.

Lähdeaineisto oli kohtalaisen laadukasta – yli puolet artikkeleista oli julkaistu julkaisukanavissa, joilla oli johtavan tai korkeimman tason luokitus. Aineistoa oli riittävästi, jotta tutkimuskysymyksiin pystyttiin vastaamaan. Ainoastaan vinoumien vähentämiskeinoista tehtyjen johtopäätösten luotettavuus on heikolla tasolla, sillä keinojen pätevyydestä aineiston artikkeleissa ei ollut riittävästi kokeellista näyttöä.

6 Yhteenveto

Tutkielmassa selvitettiin, millaisia systemaattisia ajatteluvirheitä eli kognitiivisia vinoumia ohjelmistokehittäjillä voi esiintyä ja mitä vaikutuksia niillä voi olla ohjelmistotuotannossa. Lisäksi tutkittiin, millaisia keinoja tai menetelmiä vinoumien vähentämiseen on ehdotettu tai kehitetty.

Vinoumilla on useita eri vaikutuksia ohjelmistojen suunnittelussa, toteutuksessa, testauksessa ja projektinhallinnassa. Sen seurauksena ohjelmistoon voi jäädä enemmän virheitä ja työmääräarviot olla epärealistisia, mikä voi johtaa budjetin ylittymiseen ja projektien viivästymiseen.

Vinoumista löydettiin monenlaisia vaikutuksia työmääräarvioinnissa. Tehdyistä arvioista ollaan usein liian itsevarmoja, ja arvioijien tulisi tiedostaa, että mikä tahansa arviointitilanteessa esillä oleva tai mainittu luku voi vaikuttaa työmääräarvioihin. Vinoumien vaikutus voimistuu, jos työmääräarvioinnissa ei keskitytä siihen mitä kaikkea tehtävän toteutukseen vaaditaan konkreettisesti – tehtävien laajuutta saatetaan arvioida pelkästään korkealla tasolla, jolloin arviot ovat optimistisempia kun kaikkia yksityiskohtia ei oteta huomioon.

Ohjelmistosuunnittelijat saattavat helposti yliarvioida omien ja tiimin kontribuutioiden hyötyjä ja merkityksellisyyttä. Ulkopuolelta tulevien ideoiden ja teknologioiden hyötyjä vähätellään ja siksi niitä pidetään vähemmän merkityksellisinä kuin sisäpiirissä kehitettyjä ideoita ja teknologioita. Kehitystyöhön käytetty aika ja ponnistelu hämärtävät omien tuotosten todellista arvoa; mitä enemmän vaivaa jonkin asian tekemiseen käytetään, sitä arvokkaampana se tekijänsä silmissä näyttää jopa kohtuuttoman paljon.

Kokemattomuus ohjelmistosuunnittelusta altistaa uransa alussa olevat kehittäjät vinoumille. Kokemuksen kautta kertyneen tiedon avulla ohjelmistosuunnittelijat pystyvät tuottamaan useita eri ratkaisuja ongelmiin eivätkä tyydy ensimmäiseen kehittämäänsä ratkaisuun. Kokeneet kehittäjät pyrkivät myös kyseenalaistamaan ja parantelemaan ratkaisujaan, mikä on harvinaisempaa kokemattomammilla kehittäjillä. Toisaalta kokeneet kehittäjät ovat alttiita olemaan liian itsevarmoja omista taidoista ja tiedoista.

Vinoumat voivat saada ohjelmistosuunnittelijat kiinnittämään enemmän huomiota joihinkin tiettyihin koodin tai dokumentaation osiin: usein vieraillut ja arvioidut alueet muistetaan paremmin ja niitä myös käsitellään enemmän. Tämä voi lisätä riskiä paitsioon jäävien osien vanhentumiselle ja teknisen velan syntymiselle.

Vinoumat voivat aiheuttaa myös puutteellista testausta. Ohjelmistokehittäjät haluavat varmistaa, että heidän toteuttamansa ohjelmisto toimii, joten vinoumat saavat heidät kirjoittamaan testejä, jotka vahvistavat järjestelmän toimivuuden. Testauksen tavoite on kuitenkin myös varmistaa, ettei järjestelmässä ole vikoja, joten testaajien tulisi pyrkiä tekemään testejä, jotka voisivat osoittaa puutteita ja virheitä järjestelmässä.

Koodin uudelleenkäyttö on yleensä ottaen tehokasta ja hyödyllistä, koska kaikkea

ei tarvitse tehdä alusta alkaen itse. Mutta vinoumien takia koodin uudelleenkäytössä ohjelmistokehittäjillä on taipumus luottaa koodin sopivuuteen myös uudessa kohteessa, jolloin ylimääräiset ominaisuudet tai jopa virheet voivat samalla siirtyä kohteeseen koodin mukana.

Ehdotuksia vinoumien vähentämiseksi löytyi kohtalaisen paljon, mutta kehitetyistä ja toimivista menetelmistä löytyi vähänlaisesti tutkimusta. Kognitiivisten vinoumien vähentämiseksi ei riitä, että ohjelmistokehittäjät ovat tietoisia vinoumien vaikutuksista ja tilanteista, joissa niitä voi ilmetä, sillä se on osoittautunut tehottomaksi keinoksi vinoumien tutkimuksessa.

Työmääräarviointiin sopivia vinoumien vähentämiskeinoja ovat suunnittelupokeri ankkuroitumisen estämiseksi sekä keskittyminen tehtävän laajuuteen, eli kaikkiin niihin yksityiskohtiin, joita tehtävän suorittamiseen vaaditaan. Tällöin tehtävien työmääristä tehdään realistisempia arvioita ja arviot ovat lähempänä todellisia työmääriä. Arvioiden liiallista optimismia voidaan välttää keskustelemalla arvioista yhdessä muiden kanssa, jolloin voidaan ottaa huomioon useampia tehtävien toteuttamiseen vaadittavia tekijöitä ja vaihtaa ajatuksia arvioiden takana olevista oletuksista ja perusteluista. Toinen tapa vähentää liian optimistisia työmääräarvioita on tehdä kaksi arviota: ihanteellisissa ja huonoissa olosuhteissa. Tällöin fokus siirtyy jälleen tehtävän toteuttamiseen vaadittaviin yksityiskohtiin sekä yllättäviin tapahtumiin, jotka voisivat haitata tehtävän suorittamista.

Vinoumien vaikutuksen vähentämiseen vaaditaan ohjelmoinnissa ja testauksessa kykyä tunnistaa tilanteet, joissa vinoumia tavallisesti esiintyy, sekä mielen lujuutta vastustaa tyytymistä ensimmäiseen mieleen tulevaan ratkaisuun. Omiin oletuksiin ja päätelmiin tulisi suhtautua epäilevästi ja pyrkiä etsimään niistä heikkouksia tai mahdollisia virheitä. Tällöin liiallinen itsevarmuus ratkaisujen oikeellisuuteen vähenee, ja eri vaihtoehtoja pystytään harkitsemaan objektiivisemmin. Myös testauksessa tulisi pyrkiä kiinnittämään enemmän huomiota mahdollisten ohjelmointivirheiden etsimiseen järjestelmän toimivuuden varmistamisen ohella.

Kaikkiin vinoumiin ei löytynyt niiden vaikutusta vähentäviä keinoja. Kognitiivisia vinoumia pitäisi tutkia vielä enemmän ohjelmistotuotannon näkökulmasta ja kehittää helposti toteutettavia vinoumien vähentämismenetelmiä. Vähentämiskeinojen kehittäminen vaatii kognitiivisten vinoumien syntymekanismin selvittämistä, jotta voitaisiin kehittää mahdollisimman tehokkaasti vinoumia vähentäviä keinoja. Tässä voitaisiin käyttää apuna psykologian tutkimuksen tuloksia.

Lisää tutkimusta tarvittaisiin kuitenkin siitä, miten voimakkaita vaikutuksia eri vinoumilla voi olla ja miten kova tarve niiden vähentämiskeinoille on. Jos vaikutukset eivät ole merkittäviä, eli vinoumilla ei ole huomattavia haitallisia vaikutuksia esimerkiksi ohjelmiston laatuun, olisi tehotonta lähteä kehittämään niihin vähentämiskeinoja.

Vähentämiskeinoja tulisi tutkia ensin laboratorio-olosuhteissa, jotta niiden tehokkuudesta saataisiin mahdollisimman tarkkoja tuloksia. Sen jälkeen, jos tulokset osoittautuvat lupaaviksi, keinojen tehokkuutta tulisi testata tapaustutkimuksena luonnollisissa olosuhteissa, jotta tiedettäisiin, voidaanko laboratoriotuloksia yleis-

tää tosielämän eri tilanteisiin. Keinojen ja menetelmien tulisi olla sellaisia, että ne vähentävät vinoumia ilman merkittäviä vaatimuksia itse päätöksentekijöiltä, sillä vinoumien vastustaminen ilman ulkoisia apukeinoja on tutkitusti osoittautunut hyvin vaikeaksi.

Tehtäviä ja ympäristöä muuttamalla voidaan tehdä päätöksentekijän ajatusprosesseista ja käsillä olevasta ongelmasta paremmin yhteensopivia niin, että päätöksentekijöiden luontaisia päättelykykyjä voidaan käyttää mahdollisimman tehokkaasti ilman, että vinoumia pääsee syntymään. Esimerkiksi suunnittelupokeri on tehokas ja helppo menetelmä, joka estää ankkurointivaikutusta syntymästä. Se on helposti toteutettava menetelmä eikä myöskään vaadi päätöksentekijöiltä juuri minkäänlaista opettelua tai tietoa kognitiivisista vinoumista. Menetelmä onkin juuri siksi erittäin tehokas tapa vinoumien vähentämiseksi.

Suunnittelupokeri toimii kuitenkin vain yksittäisessä projektiaktiviteetissa eli työmääräarvioinnissa. Koska eri vinoumat ilmenevät hyvin spesifeissä tilanteissa, eri kognitiivisten vinoumien vähentämiseksi vaaditaan todennäköisesti useita eri menetelmiä eri projektiaktiviteetteja varten.

Panostamalla vinoumia vähentäviin keinoihin voitaisiin parantaa ohjelmistotuotannon prosesseja ja metodeja sekä parantaa ohjelmistojen laatua, ja tuloksena pystyttäisiin toimittamaan vähemmän virheellisiä ohjelmistoja ajallaan ja budjetin mukaisesti.

Lähteet

1 M. Jackson, "The name and nature of software engineering," vol. 5316, pp. 1–38, 01 2007.

2 G. Casale, C. Chesta, P. Deussen, E. D. Nitto, P. Gouvas, S. Koussouris, V. Stankovski, A. Symeonidis, V. Vlassiou, A. Zafeiropoulos, and Z. Zhao, "Current and future challenges of software engineering for services and applications," *Procedia Computer Science*, vol. 97, pp. 34 – 42, 2016. 2nd International Conference on Cloud Forward: From Distributed to Complete Computing.

3 A. Tversky and D. Kahneman, *Judgment under Uncertainty: Heuristics and Biases*, pp. 141–162. Dordrecht: Springer Netherlands, 1975.

4 M. Razavian, A. Tang, R. Capilla, and P. Lago, "In two minds: how reflections influence software design thinking: How reflections influence software design thinking," *Journal of Software: Evolution and Process*, vol. 28, 06 2016.

5 M. Jørgensen, "The influence of selection bias on effort overruns in software development projects," *Information and Software Technology*, vol. 55, no. 9, pp. 1640 – 1650, 2013.

6 R. Nickerson, "Confirmation bias: A ubiquitous phenomenon in many guises," *Review of General Psychology*, vol. 2, pp. 175–220, 06 1998.

7 A. Wilke and R. Mata, *Cognitive Bias*, vol. 1, pp. 531–535. 12 2012.

8 J. Ehrlinger, W. Readinger, and B. Kim, "Decision-making and cognitive biases," *Encyclopedia of Mental Health*, 12 2016.

9 A. Tversky and D. Kahneman, "Judgment under uncertainty: Heuristics and biases," *Science*, vol. 185, no. 4157, pp. 1124–1131, 1974.

10 D. Kahneman, *Thinking, fast and slow.* New York: Farrar, Straus and Giroux, 2011.

11 D. Lovallo and O. Sibony, "Before you make that big decision...," *Harvard business review*, vol. 89, pp. 50–60, 137, 06 2011.

12 J. A. Marshall, P. C. Trimmer, A. I. Houston, and J. M. McNamara, "On evolutionary explanations of cognitive biases," *Trends in Ecology and Evolution*, vol. 28, no. 8, pp. 469 – 473, 2013.

13 T. Sharot, "The optimism bias," *Current Biology*, vol. 21, no. 23, pp. R941 – R945, 2011.

14 R. Mohanani, I. Salman, B. Turhan, P. Rodríguez, and P. Ralph, "Cognitive biases in software engineering: A systematic mapping study," *IEEE Transactions on Software Engineering*, pp. 1–1, 2018.

15 M. Fleischmann, M. Amirpur, A. Benlian, and T. Hess, "Cognitive biases in information systems research: A scientometric analysis," 06 2014.

16 D. Johnson and J. Fowler, "The evolution of overconfidence," *Nature*, vol. 477, pp. 317–20, 09 2011.

17 A. Tversky and D. Kahneman, "The framing of decisions and the psychology of choice," *Science*, vol. 211, no. 4481, pp. 453–458, 1981.

18 M. I. Norton, D. Mochon, and D. Ariely, "The ikea effect: When labor leads to love," *Journal of Consumer Psychology*, vol. 22, no. 3, pp. 453 – 460, 2012.

19 R. Katz and T. J. Allen, "Investigating the not invented here (nih) syndrome: A look at the performance, tenure, and communication patterns of 50 r & d project groups," *R&D Management*, vol. 12, no. 1, pp. 7–20, 1982.

20 G. Calikli and A. B. Bener, "Influence of confirmation biases of developers on software quality: An empirical study," *Software Quality Journal*, vol. 21, pp. 377–416, June 2013.

21 B. Fischhoff, "Debiasing/Kahneman, D., Slovic, P. and Tversky, A," in *Judgment Under Uncertainty: Heuristics and Biases* (D. Kahneman, P. Slovic, and A. Tversky, eds.), Cambridge University Press, 1982.

22 B. Aczel, B. Bago, A. Szollosi, A. Foldes, and B. Lukacs, "Is it time for studying real-life debiasing? evaluation of the effectiveness of an analogical intervention technique," *Frontiers in Psychology*, vol. 6, 08 2015.

23 C. Morewedge, H. Yoon, I. Scopelliti, C. Symborski, J. Korris, and K. Kassam, "Debiasing decisions: Improved decision making with a single training intervention," *Policy Insights from the Behavioral and Brain Sciences*, vol. 2, pp. 129–140, 10 2015.

24 T. Mussweiler, F. Strack, and T. Pfeiffer, "Overcoming the inevitable anchoring effect: Considering the opposite compensates for selective accessibility," *Personality and Social Psychology Bulletin*, vol. 26, pp. 1142–1150, 11 2000.

Aineisto

[A1] G. Allen and J. Parsons, "A little help can be a bad thing: Anchoring and adjustment in adaptive query reuse," in *ICIS*, 2006.

[A2] J. Aranda and S. Easterbrook, "Anchoring and adjustment in software estimation," in *ACM Sigsoft Software Engineering Notes*, vol. 30, pp. 346–355, 09 2005.

[A3] J. Parsons and C. Saunders, "Cognitive heuristics in software engineering applying and extending anchoring and adjustment to artifact reuse," *IEEE Transactions on Software Engineering*, vol. 30, pp. 873–888, Dec 2004.

[A4] E. Løhre and M. Jørgensen, "Numerical anchors and their strong effects on software development effort estimates," *Journal of Systems and Software*, vol. 116, pp. 49 – 56, 2016.

[A5] M. Jørgensen and T. Halkjelsvik, "The effects of request formats on judgment-based effort estimation," *Journal of Systems and Software*, vol. 83, no. 1, pp. 29 – 36, 2010. SI: Top Scholars.

[A6] G. Allen and J. Parsons, "Is query reuse potentially harmful? anchoring and adjustment in adapting existing database queries," *Information Systems Research*, vol. 21, pp. 56–77, 03 2010.

[A7] R. Jain, J. Muro, and K. Mohan, "A cognitive perspective on pair programming," in *Proceedings of the Twelfth Americas Conference on Information Systems*, vol. 6, p. 444, 2006.

[A8] M. Shepperd, C. Mair, and M. Jørgensen, "An experimental evaluation of a de-biasing intervention for professional software developers," in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, SAC '18, (New York, NY, USA), pp. 1510–1517, ACM, 2018.

[A9] K. Mohan and R. Jain, "Using traceability to mitigate cognitive biases in software development," *Commun. ACM*, vol. 51, pp. 110–114, Sept. 2008.

[A10] R. Mohanani, I. Salman, B. Turhan, P. Rodríguez, and P. Ralph, "Cognitive biases in software engineering: A systematic mapping study," *IEEE Transactions on Software Engineering*, pp. 1–1, 2018.

[A11] A. Tang, "Software designers, are you biased?," in *Proceedings of the 6th International Workshop on SHAring and Reusing Architectural Knowledge*, SHARK '11, (New York, NY, USA), pp. 1–8, ACM, 2011.

[A12] W. Stacy and J. MacMillan, "Cognitive bias in software engineering," *Commun. ACM*, vol. 38, pp. 57–63, June 1995.

[A13] K. Graaf, P. Liang, A. Tang, and H. Vliet, "The impact of prior knowledge on searching in software documentation," in *DocEng 2014 - Proceedings of the 2014 ACM Symposium on Document Engineering*, 09 2014.

[A14] G. Calikli and A. Bener, "Empirical analyses of the factors affecting confirmation bias and the effects of confirmation bias on software developer/tester performance," in *Proceedings of the 6th International Conference on Predictive Models in Software Engineering*, PROMISE '10, (New York, NY, USA), pp. 10:1–10:11, ACM, 2010.

[A15] G. Calikli, B. Aslan, and A. Bener, "Confirmation bias in software development and testing: An analysis of the effects of company size, experience and reasoning skills," in *Workshop on Psychology of Programming Interest Group (PPIG)*, September 2010.

[A16] G. Calikli and A. Bener, "Empirical analysis of factors affecting confirmation bias levels of software engineers," *Software Quality Journal*, vol. 23, pp. 695–722, December 2015.

[A17] L. M. Leventhal, B. E. Teasley, and D. S. Rohlman, "Analyses of factors related to positive test bias in software testing," *International Journal of Human-Computer Studies*, vol. 41, no. 5, pp. 717 – 749, 1994.

[A18] L. Leventhal, B. Teasley, D. Rohlman, and K. Instone, "Positive test bias in software testing among professionals: A review," in *Human-Computer Interaction - 3rd International Conference, EWHCI 1993, Selected Papers*, vol. 753 LNCS of *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 210–218, Springer Verlag, 1993.

[A19] R. Mohanani, P. Ralph, and B. Shreeve, "Requirements fixation," in *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, (New York, NY, USA), pp. 895–906, ACM, 2014.

[A20] R. Mohanani, B. Turhan, and P. Ralph, "Requirements framing affects design creativity," *IEEE Transactions on Software Engineering*, vol. PP, 10 2018.

[A21] O. Shmueli, N. Pliskin, and L. Fink, "Explaining over-requirement in software development projects: An experimental investigation of behavioral effects," *International Journal of Project Management*, vol. 33, no. 2, pp. 380 – 394, 2015.

[A22] A. Stefi and T. Hess, "To develop or to reuse? two perspectives on external reuse in software projects," in *Lecture Notes in Business Information Processing*, vol. 210, pp. 192–206, 06 2015.

[A23] E. Shalev, M. Keil, J. Lee, and Y. Ganzach, "Optimism bias in managing it project risks: A construal level theory perspective," in *ECIS 2014 Proceedings - 22nd European Conference on Information Systems*, 06 2014.

[A24] K. Molokken-Ostvold and N. C. Haugen, "Combining estimates with planning poker–an empirical study," in *2007 Australian Software Engineering Conference (ASWEC'07)*, pp. 349–358, April 2007.

[A25] M. Jørgensen, "Contrasting ideal and realistic conditions as a means to improve judgment-based software development effort estimation," *Information and Software Technology*, vol. 53, no. 12, pp. 1382 – 1390, 2011.

[A26] K. Moløkken-Østvold and M. Jørgensen, "Software effort estimation: unstructured group discussion as a method to reduce individual biasis," in *PPIG*, 2003.

[A27] M. Jorgensen, "Evidence-based guidelines for assessment of software development cost uncertainty," *IEEE Transactions on Software Engineering*, vol. 31, pp. 942–954, Nov 2005.

[A28] L. Gren, R. B. Svensson, and M. Unterkalmsteiner, "Is it possible to disregard obsolete requirements? an initial experiment on a potentially new bias in software effort estimation," in *2017 IEEE/ACM 10th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, pp. 56–61, May 2017.

[A29] M. Jørgensen, "The influence of selection bias on effort overruns in software development projects," *Information and Software Technology*, vol. 55, no. 9, pp. 1640 – 1650, 2013.

[A30] M. Jørgensen, "Unit effects in software project effort estimation: Work-hours gives lower effort estimates than workdays," *Journal of Systems and Software*, vol. 117, pp. 274 – 281, 2016.